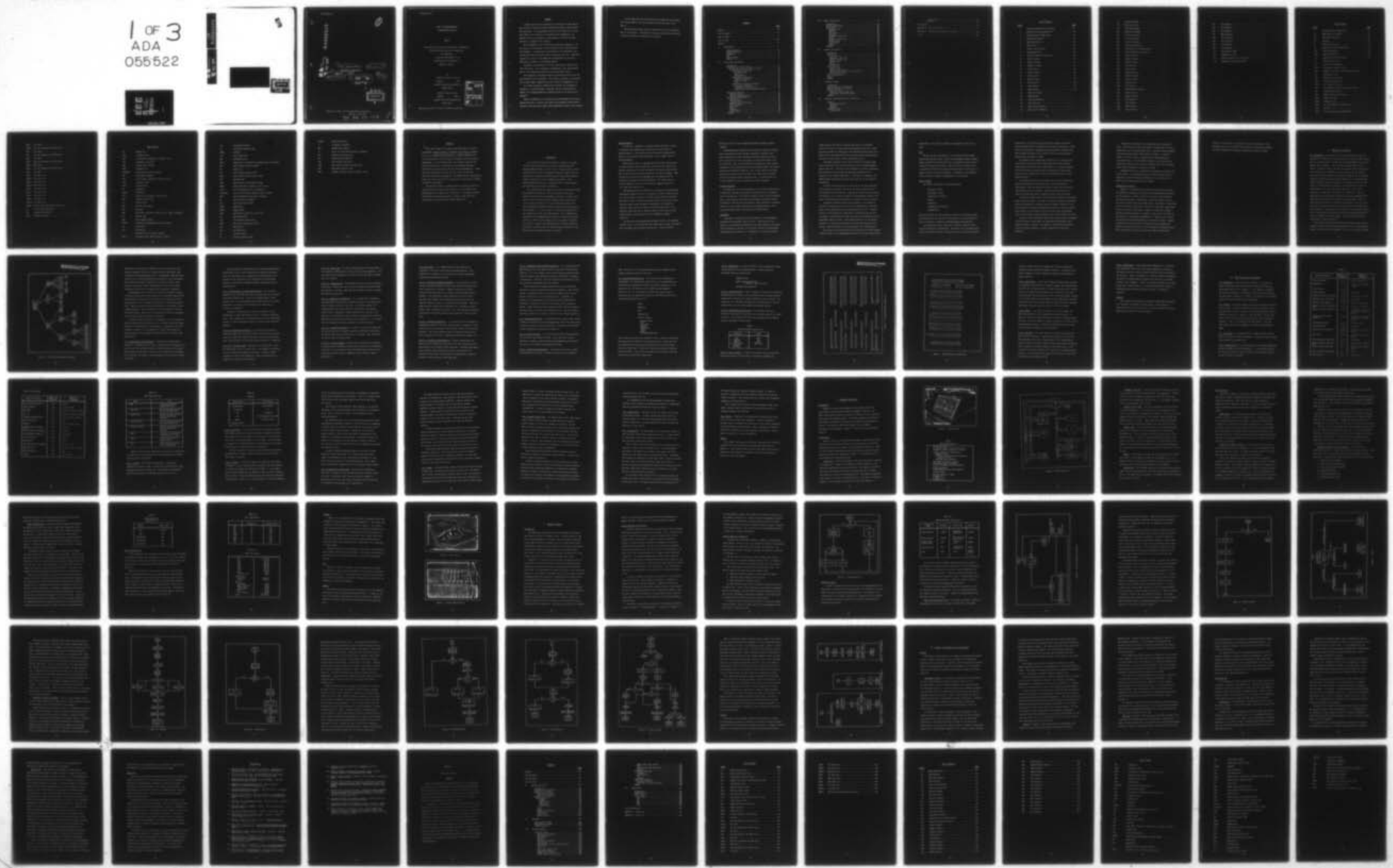
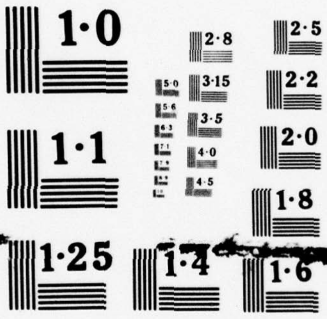




NATIONAL BUREAU OF STANDARDS
MICROCOPY RESOLUTION TEST CHART

AFIT/GE/EE/78-6

0

6 MIME: MICROPROGRAMMABLE
MINICOMPUTER EMULATOR.

11 mar 78

9 Master's THESIS,

12 224 p.

14 AFIT/GE/EE/78-6

10 Richard E. Purvis
~~Captain~~ ~~USAF~~

Ronald D. Yoho
Captain USAF

012 225

78 06 13 188

mt

AFIT/GE/EE/78-6

MIME: MICROPROGRAMMABLE MINICOMPUTER EMULATOR

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by

Richard E. Purvis, B.S.E.E.
Captain USAF
Graduate Electrical Engineering
March 1978

Ronald D. Yoho, B.S.E.E.
Captain USAF
Graduate Electrical Engineering
March 1978

Preface

A thesis such as this represents the culmination of many efforts and we wish to thank those whose contributions made our work possible and successful. It is appropriate that our first thank you be to Dr. Gary Lamont, who listened to our proposal with a sympathetic ear, agreed to serve as our advisor, and provided us with the resources necessary to accomplish this project.

We are indebted to Mr. Bob Durham, EE Laboratory Supervisor, for his help in the procurement of parts and provision of lab facilities and equipment. We appreciate the assistance of the laboratory technicians both during our thesis effort and entire AFIT tour. Particular recognition is due Mr. Dan Zambon who accomplished the electrical fabrication of MIME in an outstanding manner.

We express our gratitude to Mr. Carl Short and his staff at the AFIT school shop. The outstanding craftsmanship of Mr. Russell Murry and Mr. Tom Ersted made possible the excellent MIME cabinet.

The cooperation and professionalism evidenced by both the lab and shop personnel were outstanding. Their genuine interest in our project led to many helpful suggestions and was a great encouragement to us.

Mr. Vernon Coleman of Advanced Micro Devices, Mr. Ray Beebe of ASD/DAP, Mr. Jonathon Wallace of ASD/DAP, and Mr. Philip Carney of AFM/MJ all willingly shared their expertise with use for which we are grateful.

Equal in importance to the above was the willingness of our wives, Theresa and Vicki, to assist with some of the mundane clerical details and whose continued moral support were significant factors in our success.

We also appreciate the many constructive suggestions and comments from faculty members and fellow students and the fine work of our typist.

We have gained extreme personal satisfaction from the accomplishment of this project. The breadth of experience gained from realizing a theoretical design in hardware was invaluable.

Contents

List of Figures

List of Tables

List of Terms

AB	Address Bus
ALAT	A address latch
ALB	A less than B (Comparison of bytes A & B)
ALU	Arithmetic Logic Unit
AMD	Advanced Micro Devices
AMUX	A address mux
backplane	Wiring between edge connectors
BAR	Base Address Register
BLA	B less than A (Comparison of bytes A & B)
BLAT	B address latch
BMUX	B address mux
BP	Breakpoint
BRMUX	Branch mux
CCR	Condition Code Register (macro level)
CCU	Computer Control Unit
CS	Control Store
CSB	Control Store Buffer
DB	Data Bus
DBR	Data Buffer Register: buffers DB to D inputs of Am2901's
DC	Display code
DMA	Direct Memory Access
EPROM	Electrically programmable read-only memory
FP	Front Panel
hex	Hexadecimal
i	Subscript used to indicate integer
IMUX	Instruction mux: chooses input to 2901's

I/O	Input/Output module
IR	Instruction Register (CCU)
KYBRD	Keyboard
L.S.	Least Significant
MAB	Microaddress Bus
macro	Designation of the machine language level of operation
MAR	Memory Address Register (MEM1)
MB	Memory Bus
MBR	Memory Buffer Register (ALU)
MCCR	Micro Condition Code Register (ALU)
MDB	Microdata Bus
MEM1	Memory module: main program storage
MEMFF	Memory flip-flop: specifies read or write
micro	The microprograms level of operation
microword	Group of bits used to generate control signals
MIME	Microprogrammable Minicomputer Emulator
MR	Mask Register (in Am2914)
MLC	Micro Loop Counter (CCU)
MPMUX	Mapping mux
MPROM	Mapping PROM
MSKB	Mask Buffer: buffers PL 63-56 to DB
M.S.	Most Significant
MSP	Miscellaneous Signal Panel
MTCM	Micro Test Condition Mux (ALU)
mux	Multiplexer
N.A.	Not Applicable
PC	Program Counter
PL	Pipeline Register (CCU)

POLMUX	Polarity Mux (CCU)
Q	Q register in Am2901's
RAM	Random Access Memory
Ri	One of 16 2-port RAM locations in Am2901's
DFF	Direction Flip-flop (I/O)
SR	Status Register (Am2914)
TCM	Test Condition Mux (ALU)
TCRB	Test Condition Register Buffer (ALU)
WCR	Word Count Register (I/O)
YBFR	y Buffer: buffers output of Am2901's to DB

Abstract

This report addresses the design and fabrication of a microprogrammable, general purpose minicomputer specifically configured for use in an educational environment (AFIT's Digital Engineering Laboratory). In order to have utility as an instructional aid in the areas of computer control and microprogramming, it was determined that such a machine must have the following attributes: educationally oriented human interface, educationally oriented design, and user-microprogrammability. These high level attributes were then used as a basis for deriving functional and detail requirements. The Am2900 Bipolar Microprocessor Family was used as the basis for realizing the design which satisfied the defined requirements.

The project resulted in hardware which was used by students in digital engineering classes and laboratory to investigate microprogramming and its application to emulation. This report includes representative microcode necessary to emulate sample PDP-11/03 instructions as a demonstration of MIME capabilities.

I. Introduction

One of the most powerful and flexible techniques to emerge from the continuing evolution in digital technology has been microprogrammed computer control. The Air Force interest in this technique results from its widespread commercial adoption and its potential for simplifying problem solutions and giving equipment inherent flexibility to meet changing requirements. Thus, microprogramming has become an important facet in understanding and utilizing current digital technology.

This investigation was largely motivated by experiences gained during digital engineering class and laboratory work having computer control and microprogramming techniques as a major topic. The title for this thesis and the hardware which resulted from it was chosen both because it was an approximate mnemonic for Microprogrammable Minicomputer Emulator and because the meaning of the word mime (i.e. to act a part or to mimic) was appropriate to the capabilities of the envisioned machine. For a comprehensive discussion of microprogramming the reader is referred to the many texts (e.g. Ref 1) on the subject as well as current literature (e.g. Ref 2). However, the presentation of a few key concepts is necessary to provide a basis for understanding the purpose of the project and the approach taken by the authors.

Microprogramming

In order for a computer to accomplish data processing, a series of sequential and combinatorial events must be controlled in a specified manner. This controlling function may be accomplished by either of two principle broad techniques: use of random logic or microprogramming.

Random logic control is implemented by providing a unique group of logic circuitry which accepts each instruction, the status bits, and the clock pulses as inputs and outputs the various latch and bus gate settings as well as control pulses to the active elements. Thus, the machine instruction set implemented by the computer and the functions it performs are fixed by hardware. The lowest level of control afforded to the user is the machine language operations (i.e. add, shift, move, etc.).

Microprogramming may be defined as a technique for designing and implementing computer control as a sequence of control signals stored in a special memory as opposed to having them generated by hardware. Each group of bits is called a microword and the bits of these words represent signal states necessary to accomplish the information transfers which make up the machine instruction operations. This technique allows the user to exercise control to the level of individual register transfers which are the fundamental computer operations.

Not only does microprogramming implement control at the register transfer level, it also contrasts with random logic in that the control may be changed without hardware modification. Thus an inherent flexibility exists to meet changing requirements without hardware redesign.

Microprogramming techniques may be viewed as covering a spectrum with non-encoded and highly encoded techniques at opposite ends. Strict application of the non-encoded technique requires the micro-word to contain a bit that corresponds to each function which is to be controlled. Thus, the word tends to be lengthy but rather simple. Conversely, various microword bits may be grouped together into fields associated with specific functions. One or more levels of decoding may then be required in order to derive the actual control signals from the functional fields in the microword. In actual practice it is often expedient to use a combination of both techniques.

Problem Definition

The objective of this investigation is the design and fabrication of laboratory equipment specifically configured for use in graduate level investigations in the areas of computer control and microprogramming. Units currently available for laboratory experiments consist of commercial off-the-shelf minicomputers and microprocessor equipment. However, no unit totally combines user microprogrammability with a design philosophy meant to facilitate user understanding.

Background

The authors encountered the above problem in the Minicomputer/ Microprocessor Laboratory course, EE 6.87. This course has the objective of providing the student with an understanding of the input/ output information transfers and interfaces required in minicomputer and microprocessor operations. In addition; it is desired that the student acquire the ability to design and utilize I/O transfers. Because of a desire to learn about microprogramming and also to work with state-of-the-art technology at the integrated circuit level, the authors chose the Advanced Micro Devices 2900 Learning and Evaluation Kit (Ref 3) as the object of their laboratory project.

The Am2900 integrated circuit family consists of bipolar Large Scale Integration (LSI) devices designed for use in microprogrammed systems (Ref 4:2). The devices are cascadable in four-bit increments, thus allowing the word length and addressing capability to be tailored to the intended application. The reader is referred to the Am2900 Bipolar Microprocessor Family Data Book (Ref 4) for complete information.

In order to utilize the kit in realizing the EE 6.87 objectives, it was decided to interface one kit to act as the control unit for I/O occurring in the second kit. Although the authors were successful in causing one kit to operate under the microprogrammed control of the other, the ultimate goal of exploring I/O transfers was not achieved.

Literature provided with the kit stated that it was primarily intended to familiarize the user with the characteristics and performance of the Am2900 Family and microprogramming (Ref 3:1). However, because of the difficulties encountered in interfacing the two kits, the authors concluded the kits were not suited for investigating microprogrammed computer control. These difficulties were in the general categories of signal inaccessibility (for both monitoring and interfacing), limited displays, tedious programming, and limited memory.

Even though the ultimate goal was not realized, the authors gained valuable insight into microprogramming and how its utility might best be demonstrated. These ideas resulted in the proposal of this thesis topic.

Goal

The goal of this investigation is to provide the AFIT Digital Engineering Laboratory with a small microprogrammable digital computer having a generalized architecture and a configuration tailored specifically for educational purposes. This unit will be suited for use in laboratory instruction and student design projects in the areas of computer control and microprogramming. In addition to the hardware, a User's Manual and sample microprograms will be provided.

Plan of Attack

The effort consisted of several major phases:

Literature search

Minicomputer survey

Requirements definition

Design

Fabrication

Integration and checkout

Documentation

The first objective of the literature search was to determine whether any previous work was relevant to the problem. The second objective was to clarify the concepts of microprogramming and emulation.

The second step of the effort was to obtain information about typical minicomputer architectures. The PDP-11, Nova, and HP21MX were chosen for detailed analysis as they were three of the most frequently used machines in the military research and development environment and were representative of commercially available architectures. These three units along with a suggested design presented in Advanced Micro Devices (AMD) literature (Ref 5) served as the basis for deriving a generalized architecture for MIME.

The purpose of the requirements definition was to translate the overall goal into the technical requirements which when satisfied would provide a solution to the stated problem. A top-down approach in which each level of requirements was factored into successively lower levels was applied until there was sufficient detail to allow hardware design.

At various points during the design, trial circuits were breadboarded and analyzed to verify that they in fact fulfilled the requirements. When the design was found to be deficient, the design was iterated and then reverified. Redesign also occurred as the result of discovering invalid assumptions about module interfaces.

After first generating an overall functional design, the individual module design was begun. The order of the detail design was predicated upon the philosophy of providing a means of hardware checkout as soon as possible and retaining the greatest flexibility as long as possible. This resulted in the design of the control panel being accomplished first, the processor and memory modules next, and the control unit last.

Because of time constraints, fabrication and design were overlapped. As detail design on each module was completed, fabrication on that module was begun. Because of the degree of flexibility incorporated into the functional design, no serious problems arose from this approach.

Integration and checkout were accomplished on an incremental basis. Each module was audited for physical conformance to the documentation prior to entering module test. The module test essentially verified the proper signal flow through the module. In order to minimize isolation problems, integration was accomplished on an incremental basis. As each module was added to the mainframe, the appropriate functions were verified. Detailed module test procedures may be found in Appendix A, MIME User's Manual.

Documentation was also accomplished incrementally throughout the entire project. The documentation was required to reflect the configuration of both the design and fabrication at all times and to facilitate the understanding and use of MIME.

Organization of Thesis

Chapter I has introduced the problem and goal and has provided the background information necessary for understanding and evaluating the investigation. Chapter II and III contain the MIME functional and detail requirements respectively. These requirements served as the project specification during MIME design. Paragraphs in these chapters are numbered to facilitate correlation of the detail requirements with the functional requirements. Appendix A, MIME User's Manual, includes operating procedures, schematics, and a parts list which provide details of the hardware implementation. Chapter IV illustrates the application of microprogramming to an emulation problem. Chapter V presents the project results, conclusions, and recommendations.. Appendix B presents the details of the configuration management procedures used during the project and provides sample documentation.

Because of their volume, the wiring tables were not included in this document, but they are available at the School of Engineering, AFIT/ENG, Electrical Engineering Laboratory, Wright-Patterson AFB, OH 45433.

II. Functional Requirements

2.0 Introduction. The contents of the Problem Definition and Goal sections were formalized to succinctly state the top level MIME requirement as an educationally oriented, user-microprogrammable small general purpose digital computer. This statement included three distinct facets. In order to insure that the final product would be appropriate for an educational environment, it was necessary that its intended application be given primary consideration throughout the design phase. Secondly, providing the user with the capability to microprogram MIME entailed specific design requirements in the computer control section. Lastly, it was necessary to generalize the architecture in order to design MIME to have a general purpose capability. Thus, three categories of subordinate requirements were identified. They are educationally oriented design requirements, user-microprogrammability requirements, and generalized architecture requirements. Each of these was then factored into component requirements at the next lower level and the process repeated until the functional requirements had been identified. The purpose of this chapter is to present the development of the functional requirements that must be fulfilled by MIME in order to accomplish the project goal. The hierarchy of functional requirements is shown in Figure 1.

The Problem Definition phase specifically addressed the need for laboratory equipment specifically configured for use in investigating computer control and microprogramming. Thus, it is intended that MIME

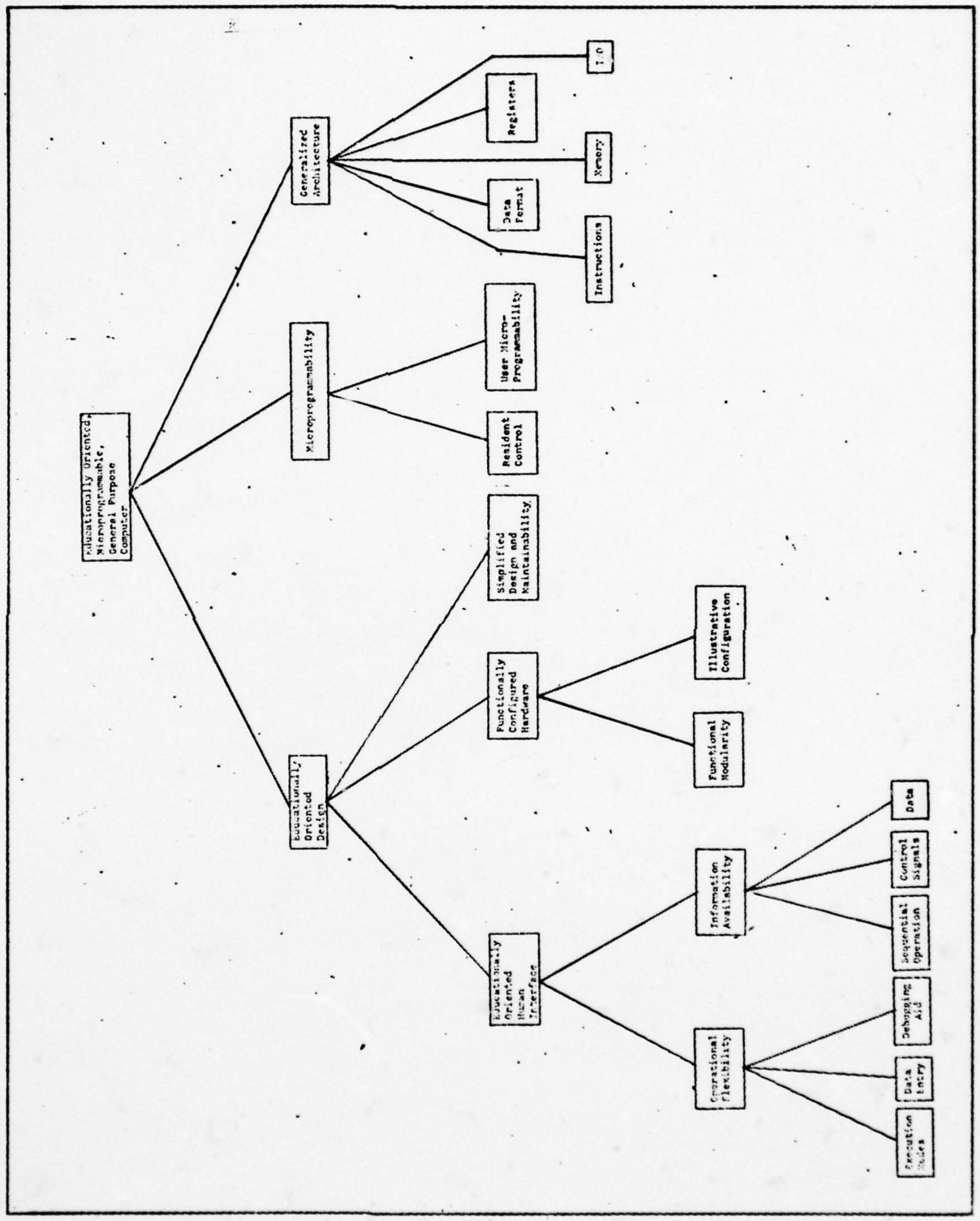

Educationally Oriented, Microprogrammable, General Purpose Computer
Generalized Architecture
Microprogrammability
Educationally Oriented Design
I/O
Registers
Memory
Data Format
Instructions
User Micro-Programmability
Resident Control
Simplified Design and Maintainability
Functionally Configured Hardware
Illustrative Configuration
Functional Modularity
Educationally Oriented Human Interface
Information Availability
Operational Flexibility
Data
Control Signals
Sequential Operation
Debugging Aid
Data Entry
Execution Modes

Figure 1. Functional Requirements Hierarchy

should have the capability to familiarize the user with basic principles of computer control and to demonstrate the applications and flexibility of microprogrammed control. Subsequently, MIME must allow the user to test his understanding by executing his own microprograms. This reflects the authors' opinion that in order for MIME to have utility in this area, it must contribute both to the user's understanding of theory and to his application of practical techniques.

Although MIME was conceived as a solution to a specific problem, it was also desired that it have potential for additional applications. The required data transfers are one facet to be considered in determining whether or not a computer is suitable for a particular application. These in turn dictate what data paths must be available in the computer hardware. Thus, the fact that it was desirable for MIME to be general purpose required that the architecture allow the implementation of an many types of transfers as possible.

In order to limit the scope of investigation, it was necessary to define the generalized architecture to be that which would allow implementation of a specified set of operations. As indicated in the Plan of Attack in Chapter I, the capabilities of three small commercial digital computers and an example from AMD literature constituted that specified set.

2.1. Educationally Oriented Design.

Regardless of whether MIME is employed as a demonstration tool or as a test-bed for microprograms, the user must be able to provide stimuli to MIME and observe the response. The structure of MIME must allow the user to visualize the interaction of the hardware and firmware, and the hardware itself must be as simple as possible.

An educationally oriented design also includes maintainability considerations such as fault isolation and component replacement. Thus, the requirements for an educationally oriented design were divided into the categories of an educationally oriented human interface, functionally configured hardware, and maintainability considerations.

2.1.1. Educationally Oriented Human Interface. The interface between the user and MIME must allow straightforward and convenient interaction between them. It must be straightforward in order that the user is not frustrated by the effort required to use MIME. Convenience is necessary so time required for data entry or retrieval is minimized.

Because of the importance of the human interface, it was designated as a major MIME function and its design was undertaken first. This approach was also advantageous in that the requirements relative to other MIME modules would be available before they were designed.

2.1.1.1. Operational Flexibility. The operational flexibility requirement addresses the need for MIME to allow the user to accomplish his purpose with minimal laboratory setup or initialization. Thus, time available for realizing learning objectives is maximized.

2.1.1.1.1. Execution Modes. Depending on his objectives, a user may employ MIME in one of several ways. For example, he may desire to run an applications program, investigate a specific control sequence, or test a microinstruction. Thus, execution modes to support these objectives are required.

2.1.1.1.2. Data Entry. In order to enhance operational flexibility, it is required that MIME provide a front panel data entry capability. The user shall have the capability to select the level and type of information to be entered.

2.1.1.1.3. Debugging Aid. Because MIME will be used in an experimental environment, features that will facilitate software and firmware debug are desirable. Therefore, MIME shall provide means to monitor software and firmware program flow.

2.1.1.2. Information Availability. It is required that information concerning MIME control signals, sequential operation, and data processing be readily available to the user. In addition, because user-microprogrammability allows MIME to be controlled at the register transfer level, commensurate signal availability is required. This information is necessary in order to present a comprehensive picture of MIME operation and allow the data flow to be monitored. It is required that this information be clearly identified to avoid confusion.

2.1.1.2.1. Sequential Operation. In order for the user to understand the sequential nature of computer operation, information identifying the machine state at any arbitrary time is required. This illustrates the logical flow of the processing.

2.1.1.2.2. Control Signals. Control Signal availability is required to demonstrate the mechanisms by which data transfers are initiated and controlled. The user must be able to determine whether failure of an attempted operation was due to improper control signals or system malfunction.

2.1.1.2.3. Data. It is required that all data locations be accessible to the user at any time during MIME operation. This information is required to allow the user to observe intermediate computation and processing results.

2.1.2. Functionally Configured Hardware. The requirement for functionally configured hardware expresses the authors' philosophy that, from an educational viewpoint, a computer system is most easily understood when functions can be uniquely associated with discrete hardware. This requirement allows advantage to be taken of the fact that most students are introduced to digital processing and computer hardware by means of a functional block diagram. These diagrams typically contain five functional units: Arithmetic, Control, Memory, Input, and Output (Refs 6:27, 7:3). This approach facilitates visualization of the data processing and transfers in terms of hardware.

2.1.2.1. Functional Modularity. It is required that the hardware be segregated according to the major function which it supports in order to facilitate the association of the hardware with the function being performed. The required modules shall be determined from the functional grouping of the detail requirements specified in Chapter III.

2.1.2.2. Illustrative Configuration. Hardware implementation of MIME functional requirements shall facilitate the visualization and understanding of the function being performed. Consistent with educational purposes, MIME configuration shall be representative of commercial architectures and functional implementations in order to illustrate current hardware design approaches.

2.1.3. Simplified Design and Maintainability. It is required that the MIME design be the least complex which can perform all the specified functions. It is the authors' opinion that unnecessary sophistication would only obscure the fundamental principles which MIME is intended to demonstrate. Therefore, only those functions which directly result from specified requirements shall be included.

The ease with which hardware malfunctions can be isolated and corrected is closely associated with design complexity. Ease of maintenance is important because it has a direct impact upon the availability of MIME for use as an educational tool. Therefore, in addition to the requirement for a simplified design, MIME shall be structured to enhance the isolation of failures to the functional module. Signal visibility and accessibility shall also be provided to facilitate fault isolation to the component level. Specific troubleshooting procedures are included in the MIME User's Manual, Appendix A.

2.2. Microprogrammability. Microprogrammability requires that control of MIME be accomplished using microprogramming rather than random logic to decode and execute machine instructions. Motivation for inclusion of this requirement was discussed in Chapter I.

2.2.1. Resident Control. It is required that MIME include provision for nonvolatile microprogram storage. Thus, MIME can be used to demonstrate computer operation under microprogrammed control without the necessity of loading a microprogram each time.

2.2.2. User-Microprogrammability. Provision for the user to apply microprogramming techniques to computer control must be included in

MIME. Therefore, it is required that MIME contain alterable microprogram storage accessible to the user.

2.3. Generalized Architecture. The initial step in defining the requirements for a generalized architecture was the determination of the functional requirements necessary to achieve equivalency to the capabilities of the PDP-11/03, Nova, HP21MX and AMD computers (Refs 8, 9, 10, 5). This analysis yielded four separate sets of characteristics and capabilities which were obtained with the aid of the following checklist (Ref 11):

- Memory
- Registers
- Data
- Instructions
- Special Features
  - Program Status
  - I/O
  - Interrupts
  - Masking
  - Protection
  - Timers
  - States
  - Microprogrammability

These separate sets were then combined to form a single set describing the required MIME capabilities and characteristics. Within this set the following broad categories of requirements which are discussed below were identified: instructions, data format, memory, registers, and input/output. Each of these categories was analyzed to yield the factors that impacted MIME architecture.

2.3.1. Instructions. In order for MIME to have an instruction capability equivalent to the target machines, it must execute the following classes of instructions:

Program Control

Double Precision/Word/Byte
Arithmetic, Logic and Shift

Word/Byte Data Transfers

2.3.1.1. Program Control. MIME is required to execute both conditional and unconditional versions of the following classes of program control instructions: Branch, Jump, Jump to Subroutine, Trap, and Skip. Consequently, it is necessary that MIME have the capability to test all signals which are used conditionally by the target machine.

2.3.1.2. Arithmetic/Logic/Shifting. The arithmetic and logic instructions required for equivalency to the target machines are listed in Table I. The required shift and rotate capabilities are shown in Figure 2 and Figure 3 respectively.

Table I

Arithmetic/Logic Capabilities

Arithmetic	Logic
Add Subtract Compare 2's Complement Increment Decrement	AND OR EX-OR Complement

2.3.1.3. Data Transfers. In order to accomplish the previously discussed instructions, several types of information transfers are

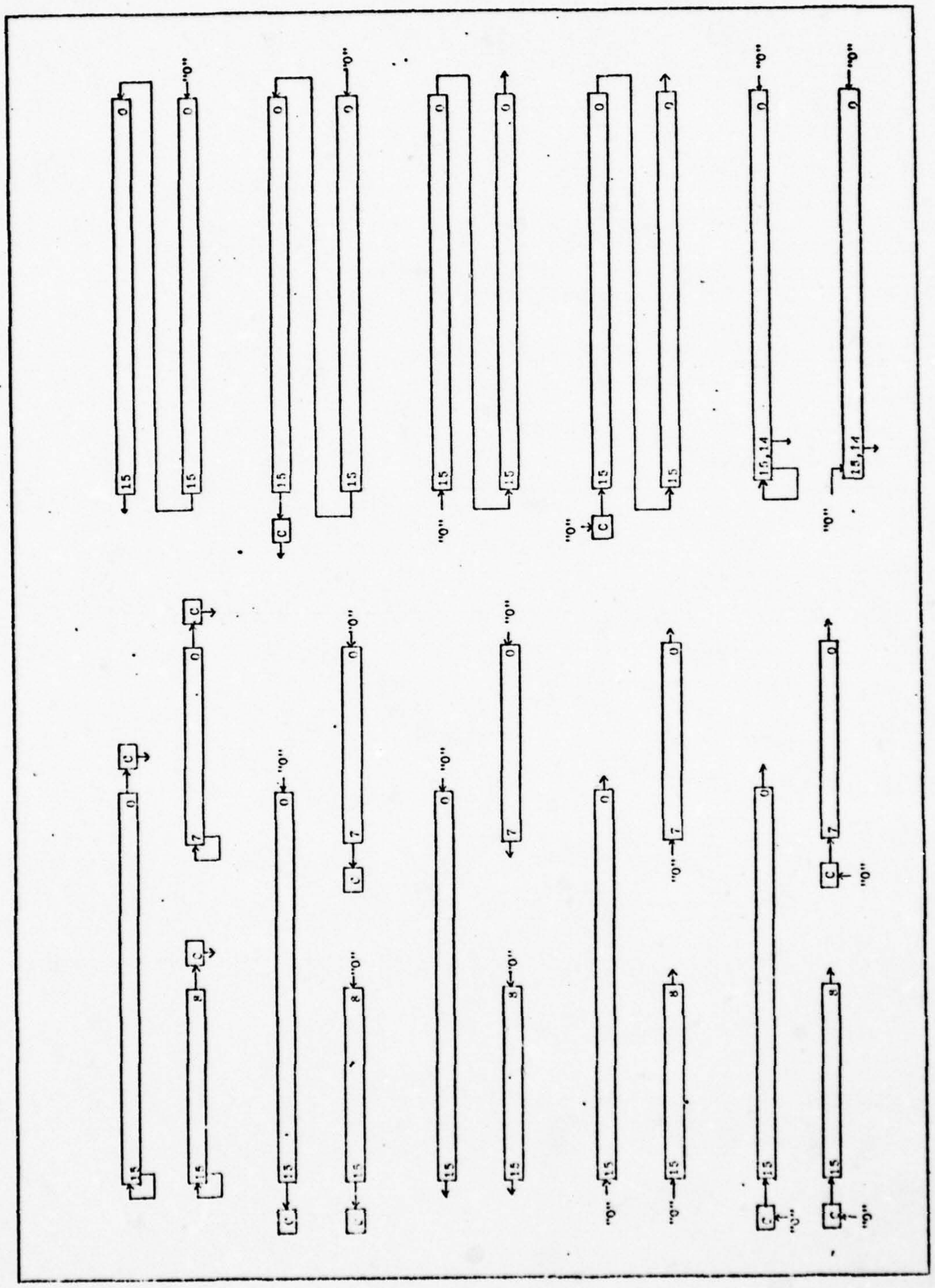


Figure 2. Required Shifting Capabilities

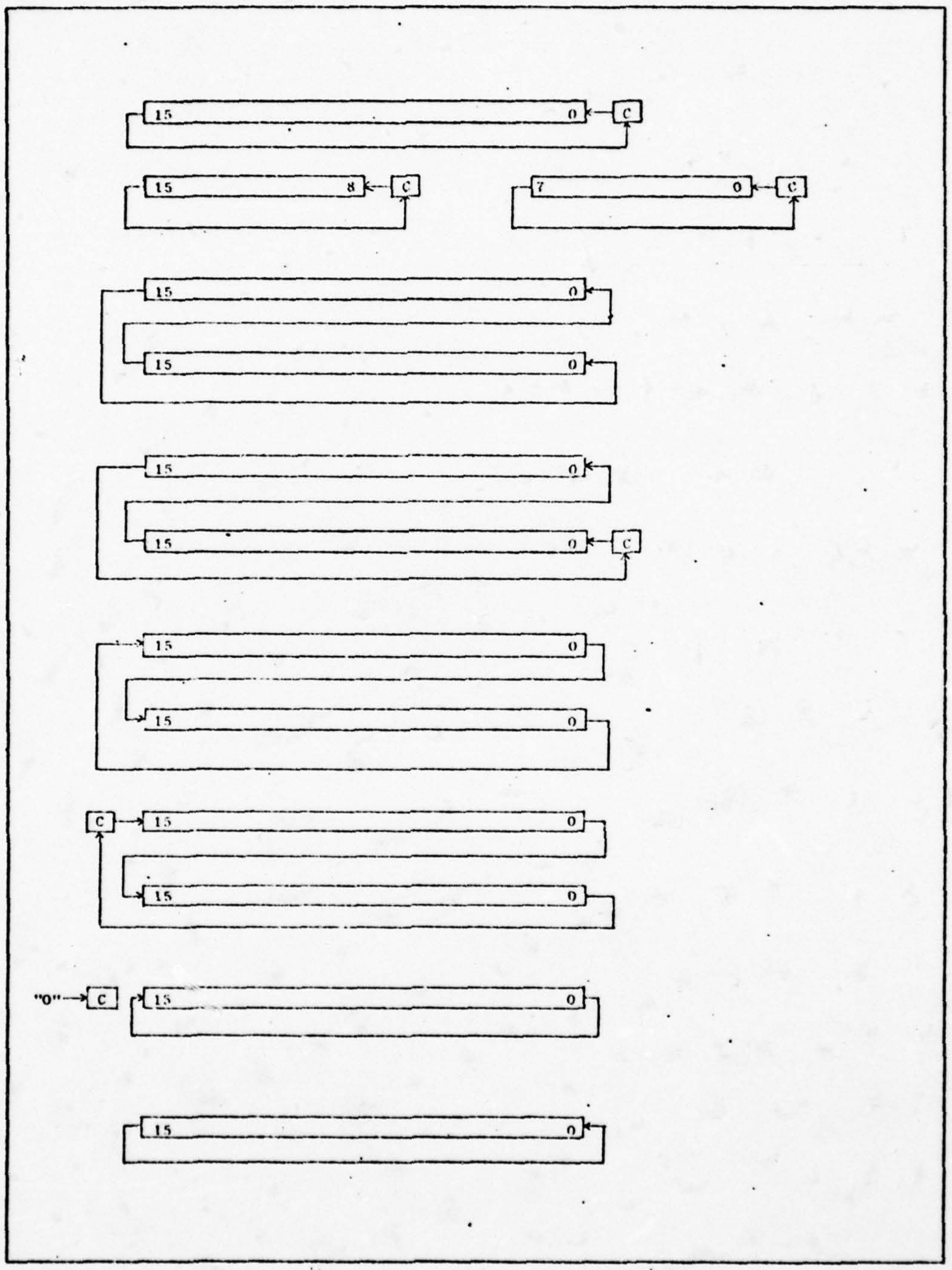


Figure 3. Required Rotate Capabilities

required. MIME must have the capability to control information transfers between itself and peripheral devices. In addition, MIME must be capable of internal register-to-register and register-to-memory transfers.

2.3.2. Data Format. A 16 bit word length and fixed point, sign plus 2's complement representation of negative numbers were common characteristics of the target machines. Because fixed point and 2's complement are a function of the instruction set capabilities, the only constraint upon MIME architecture was the 16 bit word length. Because a 16 bit word length was representative of the commercially available architectures and also provided sufficient precision for the intended application, MIME was required to have a 16 bit word length.

2.3.3. Memory.. As a result of having a 16 bit word length, the addressing capability of the target machines was 64K words. In addition, they also included the ability to address individual eight bit bytes. MIME was required to include capability for a 64K X 16 bit word addressable memeory. Pseudo byte addressability results from the byte operation capability of the processor.

2.3.4. Registers. The target machines contained both general and special purpose registers. Special purpose registers consisted of those dedicated for use as a Program Counter (PC), Memory Address Register (MAR), Memory Buffer Register (MBR), and Instruction Register (IR), accumulators, indexing registers, and stack pointer. Also included in the target machines were a maximum of six general purpose registers. It was required that registers sufficient to fulfill the same functions be included in MIME.

2.3.5. Input/Output. Requirements under paragraph 2.1.1. specified that comprehensive data entry and display capability be included in the human interface in order to provide flexibility and convenience to the operator. However, paragraph 2.1.1. did not specify the capability desirable for loading complete programs into MIME using an assembly or higher order language. In order to provide the capability of interfacing with external devices, it is required that MIME be capable of performing DMA and vectored priority interrupt I/O information transfers.

Summmary

This chapter has presented the functional requirements determined to be requisite for meeting the project goal. These functional requirements will be used as the basis for the development of detail requirements presented in the next chapter.

III. Detail Functional Requirements

3. Introduction. The purpose of this chapter is to provide the detail requirements which were derived to satisfy the functional requirements identified in Chapter II. This chapter is organized by functional categories in consonance with the functional modularity requirement of Chapter II. Table II illustrates the relationship between the functional and detail requirements.

3.1. Bussing. The target machines included both single and multiple bus structures. Because the required information transfers could be implemented in either structure, the architectural generalization imposed no constraints. However, it was the authors' opinion that separate busses would facilitate visualization of the information transfers. Thus, it was required that MIME include a Data Bus (DB), Address Bus (AB), Microaddress Bus (MAB), and Microdata Bus (MDB).

3.2. Front Panel. The requirement for a human interface shall be met by the MIME Front Panel (FP) module. It shall have the following characteristics and capabilities.

3.2.1. Debugging Aid. The FP shall provide a stand-alone capability without dependence on other MIME modules. It was desired that the FP always be available to the user without the necessity of having resident software or firmware.

Table II

Detail Requirements	Chapter III Paragraph	Chapter II Paragraph
AB, DB, MAB, MDB	3.1.	2.1., 2.1.3., 2.3.
Handwired FP	3.2.1.	2.1.1.1.3., 2.1.2.1., 2.1.3.
Power Control	3.2.2.	
Operating Modes	3.2.2.	2.1.1.1.1.
Hexadecimal Micro Data Entry	3.2.3.	2.1.1.1.2., 2.1.2.3.
Hex/Octal Macro Data Entry	3.2.3.	2., 2.3.2.
Register Load Capability	3.2.3.	2.1.1.1., 2.1.1.1.2.
Memory Location Load Capability	3.2.3.	2.1.1.1., 2.1.1.1.2.
Register Display Capability	3.2.4.	2.1.1.1.3., 2.1.1.2.2. 2.1.1.2.3., 2.1.1.2., 2.1.3.
Memory Location Display Capability	3.2.4.	2.1.1.1.3., 2.1.1.2.2., 2.1.1.2.3., 2.1.1.2., 2.1.3.
FP Display Registers	3.2.4.	2.1.1.2., 2.1.3.
State Indicators	3.2.4.	2.1.1.2.1.
Condition Code Indicators	3.2.4., 3.3.	2.1.1.1.3., 2.1.1.2., 2.1.3.
Test Points	3.2.4.	2.1.1.1.3., 2.1.1.2., 2.1.3.
Macro Breakpoint Registers	3.2.4.	2.1.1.1.3.
Word/Byte/Double Precision ALU	3.3.	2.3.1., 2.3.2., 2.3.3.
General Purpose Registers	3.3.	2.1., 2.1.3., 2.3.1., 2.3.4.
Auxiliary Shifting Register	3.3.	2.3.1.
Data Transfers	3.3.	2.1., 2.3.1.

Table II (Continued)

Detail Requirement	Chapter III Paragraph	Chapter II Paragraph
Microword Mask/Constant	3.3.	2.3.1.
Memory Capacity	3.4.	2.3.2., 2.3.3.
MAR, MBR	3.4.	2.3.3., 2.1.2.1., 2.1.2.2.
PC, IR	3.5.	2.1.2.1., 2.1.2.2., 2.3.4.
Decoding and Sequencing	3.5.	2.1., 2.1.2.1., 2.3.1.
Clock Generation	3.5.	2.3.
Mapping	3.5.	2.1., 2.1.1.2., 2.3.
Microinstruction Capabilities	3.5.	2.3.1.
Microinstruction Pipelining	3.5.	2.1.2.2.
Microbreakpoint Register	3.5.	2.1.1.1.3.
User Writable CS	3.6.	2.2.2.
Read Only CS	3.6.	2.2.1.
Vectored Priority Interrupt	3.7.	2.3.1., 2.3.5.
WCR, BAR, DFF	3.7.	2.1.2.2., 2.3.1., 2.3.5.
MIME Portability	3.8.	2.1.
Chassis	3.8.	2.1., 2.1.3.

Table III

MIME Operating Modes

Mode	Use
RUN	Continuous automatic program execution
Auto Step	Execute macroinstructions at an observable rate
Single Step	Execute one macroinstruction on each manual actuation
Auto Microstep	Execute microinstructions at an observable rate
Single Microstep	Execute one microinstruction on each manual actuation
Pause	Interrupts clock without changing state of MIME
Reset	Reinitialize MIME for program start
Halt	Halts program execution in the fetch state

Because the FP could be independently verified to be operational, it could be used in the checkout of other MIME modules and troubleshooting would be simplified.

3.2.2. Control. The FP shall provide means to control the application of power to MIME. It shall also allow selection of the operating modes given in Table III. These modes were required to provide the operational flexibility specified in 2.1.1.1.1.

Table IV

Displays

Macro Level	Micro Level
All Registers	Micro Address
ALU Output	CS
DB	PL Register
AB	Microbreakpoint Register
Memory	Keyboard Entry
Keyboard Entry	

3.2.3. Data Entry. Micro level data shall be entered in hexadecimal format in order to minimize the number of characters required for a microword. The user shall have the option of entering macro level data in either hexadecimal or octal format. An octal format is required for consistency with the target machines, but the hexadecimal format is required for future flexibility.

So that the operator is given as much flexibility as possible, the capability to load any register or memory location from the front panel is required.

3.2.4. Displays. In order to support the functional requirements of Chapter II, the FP shall provide the operator with the capability to selectively display the items listed in Table IV. This comprehensive display capability is required in order to provide visibility for all facets of internal MIME operations and facilitate malfunction identification. However, it is also desirable to limit the complexity of the FP design and limit the quantity of information to that which can be easily assimilated by the operator. Thus, it is required that the FP include only one display register each for the macro and micro level.

The FP shall provide indicators which identify to the operator the source of the current display or the destination of the keyboard entry. This is necessary in order to avoid confusion between various parameters.

For display purposes, two of the general purpose registers shall be designated as breakpoint registers. This provision is a debugging aid which allows the operator to halt MIME when the contents of the PC and breakpoint register match. Although any of the general purpose registers could be used, two shall be designated for this purpose as a convenience to the operator. Indicators to advise the operator of the machine state and arithmetic condition codes are also required in order to provide the user with information concerning the processing status.

In order to enhance debugging capability as well as routine monitoring of MIME operations, it is required that any control signals not displayable on the front panel be accessible via test points for logic analyzer or oscilloscope display. These signals shall include clock, synchronization, and interrupt request signals.

3.3. Arithmetic Logic Unit (ALU). The processing requirements identified in Chapter II shall be fulfilled by the Arithmetic Logic Unit (ALU). This module shall contain a processor capable of performing 16 bit word, 8 bit byte, and double precision operations in accordance with paragraph 2.3.1. requirements.

The target machine processors include such special purpose registers as accumulators, index registers, and stack pointers, and as many as six general purpose registers. However, because each target machine has a different set of these special purpose registers, greater flexibility is obtained by using general purpose registers. General purpose registers are also required for the microprogrammer's use without disturbing the contents of registers currently in use of the macro level. Therefore, the ALU shall include 16 general purpose registers and 1 auxiliary shifting register.

It shall be possible to transfer data between the DB and general purpose registers and to obtain operands from the DB or registers. Results of operations shall be stored in the ALU registers or output directly to the DB. This allows the greatest possible flexibility for the user. The ALU shall also be capable of gating 16 bits of the pipeline onto the DB for use as a macro level constant or mask.

The ALU shall make available flags that indicate the results of arithmetic operations. These flags may be used by the Computer Control Unit as the basis for conditional branches. The ALU shall accept control signals which determine the type of operation, operand source(s), and result destination.

3.4. Memory. The Memory module shall provide for main program storage. It shall have a maximum capacity of 64K x 16 bit words. The module shall include a Memory Address Register (MAR) to latch the address of the memory location being referenced and allow either synchronous or asynchronous operation. The module shall also include a Memory Buffer

Register (MBR) to isolate the memory and DB from each other. The MBR serves to protect the memory and also facilitates the illustration of information flow during a memory reference. The MAR and MBR shall be implemented as discrete hardware registers to facilitate illustration of transfers and to allow hardwired signal accessibility. In order to simplify the module circuitry, the memory shall be static type random access storage.

3.5. Computer Control Unit. The Computer Control Unit (CCU) module shall exercise overall control over MIME operations and shall provide signals to control all information transfers. The CCU shall contain the Program Counter (PC) and Instruction Register (IR). The PC is required to hold the address of the currently executing instruction and the IR to contain that instruction. The PC and IR shall be able to act as either a source or destination for data relative to the DB. These registers shall be implemented as discrete hardware registers to facilitate illustration of their function and to allow hardwired signal accessibility.

The CCU shall provide decoding of macroinstructions, microinstruction sequencing, and control signal decoding. It shall also generate master clock pulses.

In order to provide flexibility, the CCU shall be able to map macroinstruction operation codes into microroutine starting addresses. Microinstruction sequencing capabilities of conditional branching, conditional looping, and four-level conditional nested subroutining are required to illustrate representative microprogramming techniques. To illustrate the pipelining technique of overlapping the fetch and execution cycles, the CCU shall include first-level pipelining of microinstructions (Ref 12).

As a debugging aid to the microprogrammer, the CCU shall include a micro breakpoint register and provision for halting MIME when the microaddress matches the register contents.

3.6. Control Store. The Control Store (CS) module shall provide microprogram storage. It shall include both user-writable and read-only memory to allow user microprogrammability and resident control respectively. An input/output buffer shall interface the microdata bus with the CS.

3.7. Input/Output. The Input/Output (I/O) module shall handle all data transfers between MIME and external devices. It shall include an Input/Output Buffer (IOB) register which shall be used as the data interface with external devices.

The I/O module shall provide eight levels of vectored priority interrupt. Because MIME is intended for use in the educational environment, the authors do not foresee any occasion which will require more than eight levels of interrupt priority. In addition, the technique for handling more than eight levels remains essentially the same. Thus, the authors feel that eight levels provide sufficient capability and adequately demonstrate interrupt driven I/O. Therefore, MIME shall include provision for handling up to eight levels of vectored priority interrupts.

The I/O module shall notify the CCU of pending interrupts and identify the highest priority device requiring service by providing an interrupt vector. The module shall provide the capability for interrupt nesting and selective interrupt marking. It shall be possible to display and load the registers containing the mask and status information. This provides the user access to the fundamental components of I/O processing.

The I/O module shall include a Word Count Register (WCR), Base Address Register (BAR), and Direction Flip Flop (DFF) for use in accomplishing DMA (I/O) transfers.

3.8. Chassis. MIME shall be self-contained, requiring only connection to 110 vac, 50-60 Hz for operation. It shall be housed in a portable cabinet configured to allow access to all internal components for troubleshooting and replacement. These requirements provide the flexibility needed for use in the educational environment. Cooling shall be provided by a chassis mounted fan.

Summary

This chapter has presented MIME detail requirements by functional category which were derived from the functional requirements of Chapter II. The derivation may be traced using Table II and Figure 1. Hardware was then selected to implement the detail requirements as discussed in the next chapter.

IV. Hardware Realization

Introduction

Chapters II and III have presented the functional and detail requirements respectively which MIME was designed to fulfill. The detail requirements presented in Table II of Chapter III were then used as the basis for hardware design. This chapter presents an overview of the hardware realization and discusses the significant implementation choices. The reader is referred to Appendix A for a complete description of the hardware. Figure 4 illustrates the completed MIME hardware.

Architecture

Figure 5 contains a functional block diagram illustrating the MIME architecture which resulted from the functional grouping of the detail requirements described in Chapter III and summarized in Table II. MIME is a bus oriented, functionally modularized digital computer having the general characteristics summarized in Table V. The implementation of each module will be discussed briefly.

Front Panel. The FP was designed to meet the requirements specified in paragraph 3.2. Because of the quantity of parts required, it was physically implemented on two circuit cards. The displays and controls which form the cabinet front panel along with some logic circuitry were mounted on a circuit card designated FP1. The remaining components were mounted on a circuit card designated FP2. The reader is referred to Appendix A for a complete description of front panel operation.

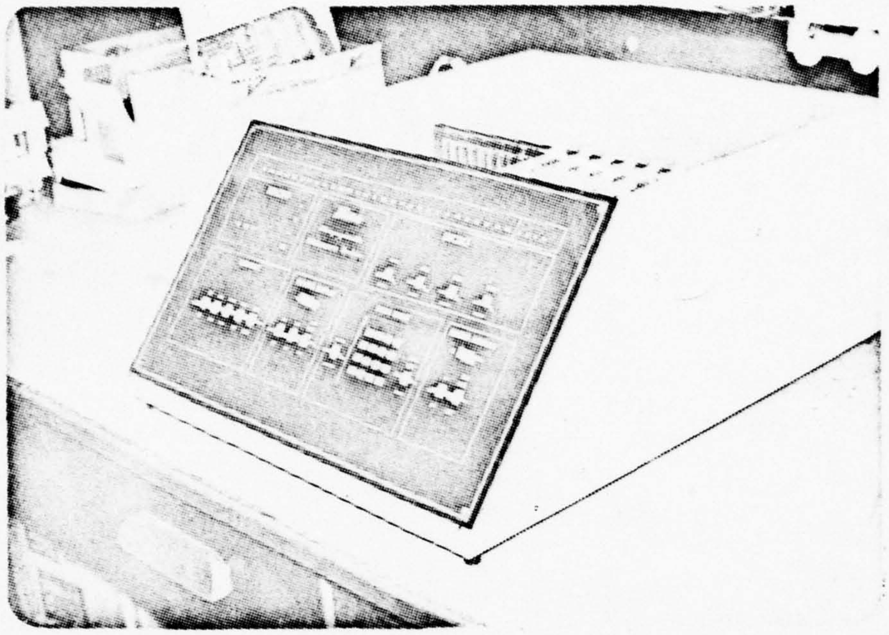


Figure 4. MIME Hardware

Table V

MIME Characteristics

16 bit data word
16 general purpose registers (16 bits)
10 special purpose registers (16 bits)

machine instruction capabilities
  program control
  double/precision/word/byte arithmetic,
    logic, and shift
  word/byte data transfers

64K words main memory capacity
synchronous memory references
8 levels vectored priority interrupt

64 bit microword
4K microwords Control Store capacity

1.43 MHz clock

110 vac, 50-60 Hz, 2 amps
21 amp @ 5 v
1.6 amp @ -5 v
2.4 amp @ 12 v

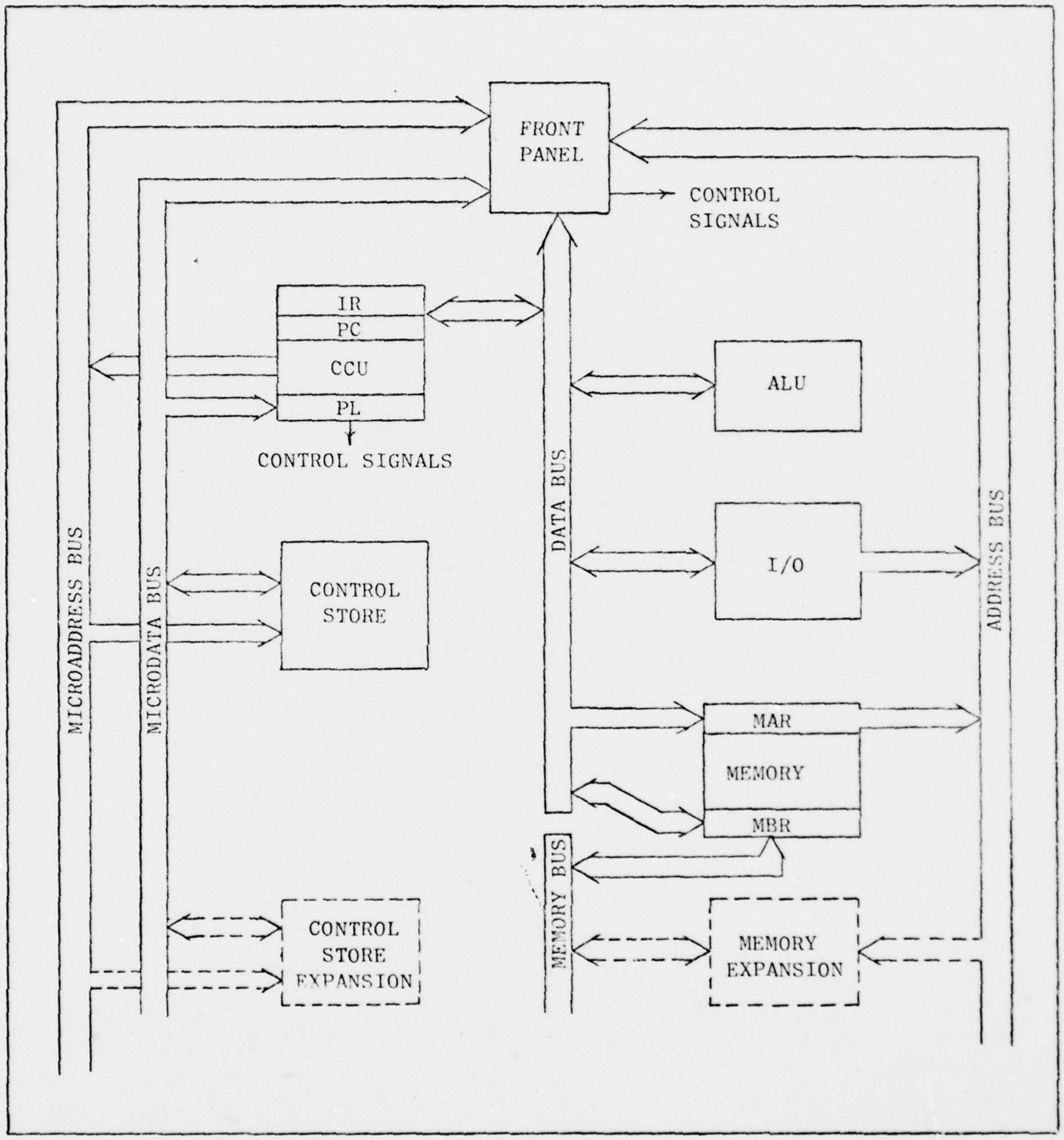


Figure 5. MIME Architecture

Arithmetic Logic Unit. The ALU circuit card contains all the circuitry necessary to meet the specifications of paragraph 3.3. The ALU is built around four Am2901 microprocessor chips (Ref 4:5-16) and includes various multiplexers and registers. It incorporates lookahead carry using the Am2902 (Ref 4:17-18).

Computer Control Unit. The CCU was designed to meet the specifications of paragraph 3.5. This module was also implemented on two circuit cards which were designated CCU1 and CCU2. The heart of the CCU capability results from the use of the Am29803, Am29811, Am2909, and Am2911 (Ref 12:2-2 to 2-21). These components provide the decoding, decision making, and sequencing capabilities required for the CCU.

Control Store. The CS module provides the microprogram storage specified in paragraph 3.6. and is contained on one circuit card designated CS1. It also includes the circuitry necessary for micro-address decoding. It consists of both RAM and EPROM storage for the reasons discussed in the detail requirements. In anticipation of CS expansion (See Recommendations, Chapter VI), this card was designated CS1, and an appropriate card rack position was left vacant for a second CS card.

Memory. The MEM module provides the main program storage for MIME and consists of RAM elements as specified in paragraph 3.4. It is designated MEM1 in anticipation of expansion and contains the circuitry necessary to decode addresses and implement the MAR and MBR.

Input/Output. The I/O module contains the BAR and WCR and provides eight levels of vectored priority interrupt as specified in paragraph 3.7. for use in accomplishing I/O transfers. The Am2914 (Ref 4:66-73) provides the interrupt capability.

Parts Selection

There were a number of times during the design phase when more than one specific hardware implementation appeared capable of satisfying the specified requirements. When this occurred, it was necessary to determine the most desirable alternative. This section summarizes those choices deemed significant and the criteria applied in the decision process.

Am2900 Family. It would have been possible to consider microprocessor families other than the Am2900 series. However, the choice of the Am2900 integrated circuit family as the basic building blocks of MIME was the result of the following considerations: As indicated in Chapter I, the authors were familiar with it from a previous project. More significantly, the Am2900 family represented commercially available state-of-the-art technology, provided the most powerful capabilities available in microprogrammable microprocessors, and illustrated the use of bit-slice technology.

Control Store Memory Selection. In keeping with the requirement for flexibility specified in paragraph 3.6., the CS included both read/write and read-only storage. This approach allowed for both user experimentation and a nonvolatile resident control function.

Static type random access memory was chosen for the read/write portion of CS in order to minimize support circuitry required for dynamic RAM (Ref 13:3-19). Even though a read-only capability was required, it was still desirable to have the capability to alter the resident control or to implement additional instruction sets. Thus an electrically programmable, ultraviolet erasable, Programmable Read-Only Memory was chosen as the read-only portion of CS. It was also desired that the EPROM be compatible with a Read-Only Memory (ROM) to allow substitution of ROM as an optional means of decreasing CS access time.

It was determined from commercial data that a representative instruction set required approximately 1K of CS for microprogrammed implementation (Ref 10:1-1). Thus, MIME was initially provided with 1024 words of EPROM storage in CS.

Because it was the authors' opinion that the writable CS would be used primarily for experiments and debugging, it was decided to provide only 256 words initially. Additionally, because CS is manually loaded from the front panel keyboard in hexadecimal format as specified in paragraph 3.2., it was felt that no more than 256 words would be utilized.

Selection of the actual memory chips involved tradeoffs among access time, bit density, pin configuration, and cost. It was necessary that the access time be sufficient to meet the overall timing goals discussed under Clock Considerations. Bit density and pin configuration were significant factors in minimizing the required circuit card area and it was desired to obtain the lowest feasible cost per bit.

Component Mounting and Interconnection. Wire wrap circuit boards with built-in IC sockets were used for all cards except FP1. These were mounted in a card rack and interconnected using edge and cable connectors. Augat 8136-URG5 cards were chosen for the following reasons:

1. Convenient IC replacement.
2. Built-in VCC and ground planes.
3. Large capacity edge connectors.
4. Rugged construction.
5. Available mounting rack.
6. Available extender card.

These characteristics were deemed desirable because of the one-of-a-kind nature of MIME and its intended laboratory use.

Clock Considerations. Information transfers between MIME modules are synchronously controlled using a single phase symmetrical clock. The decision to utilize synchronous transfers was considered carefully by the authors. Asynchronous transfers would have allowed MIME to operate at the fastest speed possible by basing the clock frequency on the fastest module and skipping clock cycles to accomodate the slower modules. A disadvantage of asynchronous operation was that it required additional logic to control the clock.

Synchronous operations base the clock frequency on the slowest operation which results in a simpler but slower machine. Table VI lists approximate speeds of the MIME modules which impacted this decision. The CCU/CS entries are the sum of CCU next microaddress computation time and CS access time. Thus, each CCU/CS entry is the total time required to compute the address of and fetch the next microword. This time is the minimum clock cycle utilizable with the particular CS implementation.

Based on this information, the fastest possible operation mode is achieved with a ROM CS implementation and asynchronous memory references. The slowest mode is achieved using synchronous operation with EPROM CS implementation. Approximate microinstruction execution rates are 3.33 MHz and 1.43 MHz respectively. The EPROM CS implementation was chosen and the substitution of ROM left as a potential enhancement. The required clock was therefore determined to be 1.43 ± 0.05 MHz. Provision for asynchronous control was included in the CCU to provide flexibility. This approach provided the simplicity desired for an educational environment and also the potential for future increases in operating speed.

Table VI

Module Cycle Time
(Approximate)

Modules	Cycle Times
ALU	300 ns
MEM	400
CCU/CS(EPROM)	675
CCU/CS(RAM)	475
CCU/CS(ROM)	300

Power Considerations

Power requirements were determined for each circuit card individually by summing the typical power requirements for each IC. All IC's required a VCC of plus 5 volts except the EPROM's which also required minus 5 and plus 12 volts. The voltage and current requirements are given in Table VII.

Power was distributed to each circuit card in the card rack (see Chassis section which follows) via the edge connectors and wire-wrap wire. No information concerning the current carrying capacity of 30 gauge wire-wrap was available so the authors experimentally determined that a single wire 12-18 inches in length could carry .5 amp over an extended time period without discernable heat build-up. The number of power connections to each card was then determined using .5 amp as the wire capacity and the requirements listed in Table VII.

Table VII

Power Requirements

	Voltage (v)	Current (amp)
FP2	5	2.0
CS1	5	1.5
CS1	-5	1.62
CS1	12	2.34
CCU1	5	1.9
CCU2	5	1.9
ALU	5	1.55
MEM1	5	6.0
I/O	5	2.05
FP1	5	4.3

Table VIII

MIME Cost Summary

FP1	$ 668.00
FP2	243.00
ALU	295.00
MEM1	207.00
CS1	380.00
CCU1	250.00
CCU2	243.00
I/O	217.00
Wire	75.00
Extender Cord	58.00
Fabrication	
Electrical	3,200.00
Chassis	680.00
Chassis	
Card Rack	48.00
Edge Connectors	75.00
Ribbon Cable	10.00
Connectors	35.00
Fan	15.00
Miscellaneous	25.00
Total	$ 6,724.00

Chassis

Figure 6 is a representation of the chassis configuration which was designed to satisfy the requirements of paragraph 3.8. The circuit card positions in the card rack are illustrated in Figure 7. All circuit cards are rack mounted except for FP1 which is directly attached to the hinged cabinet front panel. The optimum angle for the front panel is such that it is perpendicular to the user's line of sight (Ref 14:42). The actual front panel slope was determined experimentally based on anticipated bench top operation and the viewing angle for an individual of medium height.

The area at the rear of the chassis is reserved for installation of power supplies which are not yet available. The cabinet was vented and provision made for a chassis mounted fan in order to assure adequate cooling.

Cost

Table VIII contains a breakdown of the estimated costs for MIME. The estimate for each circuit card includes the cost of the card and the components mounted on it. The estimated total costs for wire and fabrication are also included in Table VIII. Fabrication is divided into two categories: electrical and cabinet.

Summary

This chapter has presented a brief description of the MIME hardware that resulted from the design and fabrication effort. In addition, the areas of significant design decisions have been described. Appendix A contains the complete description of the hardware and operating procedures.

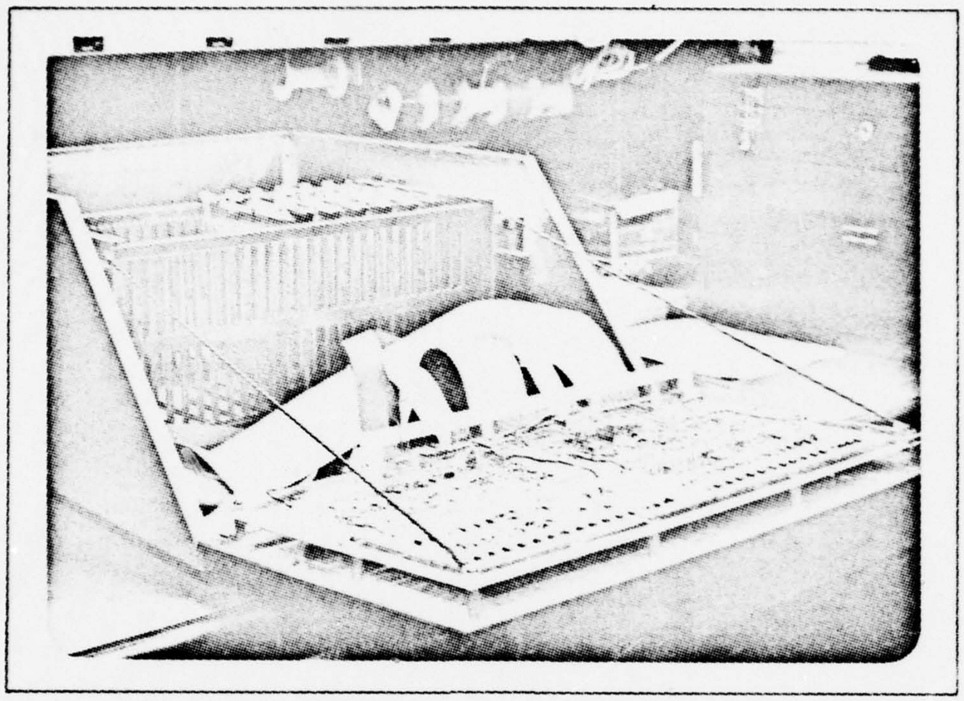


Figure 6. MIME Chassis

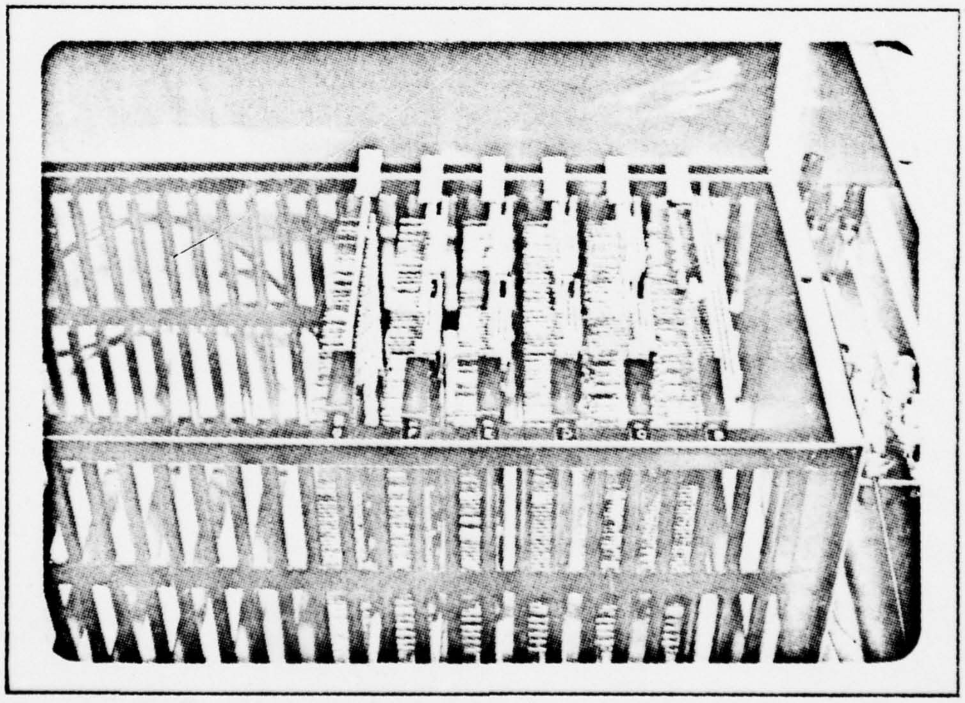


Figure 7. Circuit Card Positions

V. Emulation Example

Introduction

One significant application for microprogramming is emulation, which may be defined in a number of ways. It may be defined as the use of microprogramming to reproduce some of the hardware functions of the target machine not present on the host machine (Ref 15:265). Alternatively, emulation may be defined as the ability of one machine to execute the machine language instructions of another machine (Ref 16:15), or the combined hardware/software interpretation of the machine instructions of one machine by another (Ref 6:475).

Emulation is of interest in the military environment because of its potential for achieving software compatibility among different machines and mitigating the effects of rapid hardware obsolescence (Ref 17:29). The significance of emulation relative to the educational purposes of this project is the demonstration of the concepts of microprogramming applied to emulation. Although execution timing equivalency would also be required for a valid emulation in a realtime environment, it is not necessary for demonstration of the concept. Thus, the net effect of emulation would be the capability of running the target machine language programs on another machine with identical processing results. Therefore, for the purposes of this project, emulation has been defined as the ability of MIME to execute the machine instructions of another machine in a way such that the processing results are equivalent. The remaining sections of this chapter discuss the relationship between emulation and microprogramming, a general approach to emulation, and a specific emulation example.

Microprogramming and Emulation

A specific sequence of operations are necessary in order to execute each machine language instruction of a program. First, the instruction must be fetched from the memory. Then the op code portion of the instruction must be decoded to determine the desired operation. Finally, the control signals necessary to accomplish the required transfers must be generated. The first two steps, fetch and decode, are essentially identical for all instructions of a given machine. The final step, control signal generation, is unique for each particular instruction. In a microprogrammed machine such as MIME, fetch and decode are controlled by a microprogram executive which branches to the particular micromodule (series of microinstructions) required to control the transfers of each machine level instruction. Thus, each machine instruction is implemented as a micromodule at the register transfer level.

In order to emulate a particular instruction set, the same three steps are necessary: fetch, decode, and execution. Emulation is possible only if the emulator hardware is capable of fetching and decoding the instruction controlling transfers within its architecture to arrive at results equivalent to those of the target machine. Thus, the first requisite for emulation is the availability of a versatile architecture, one which is in fact capable of the necessary decoding and processing.

Given this, one must devise a method of controlling the required register transfers in a flexible manner. As discussed in Chapter I, a microprogrammable machine such as MIME has an inherent flexibility at the register transfer level. This gives the microprogrammer complete control of every information transfer possible within the emulator hardware and makes it possible to emulate equivalent register transfers of the target machine. Thus, the utility of microprogramming applied to emulation is readily demonstrated.

General Approach to Emulation

Assuming that the emulator hardware is capable of emulating the target machine, the first step is to specify the high level microprogram flow necessary for instruction fetch, decode, and execution. The example shown in Figure 8 includes interrupt I/O handling as specified in paragraph 3.7.

The next step is to express the fetch, decode, and interrupt handling routines in Register Transfer Language (RTL). An example of this is included in the PDP-11/03 example at the end of this chapter.

Next, each machine instruction must be described in RTL. The authors found the following three step process useful:

1. Read instruction description in target machine handbook.
2. Represent target machine transfers in RTL.
3. Represent emulator transfers in RTL.

The most difficult step is the transition from target machine RTL to emulator RTL. There is, not necessarily a one-to-one correspondence between the transfers. The end result of the processing must be the same, but the intermediate steps may be different. This process must be repeated for each instruction of the target machine.

As the last step, the RTL for the emulator is converted to appropriate microcode. This, of course, may be done incrementally as each instruction is specified in RTL.

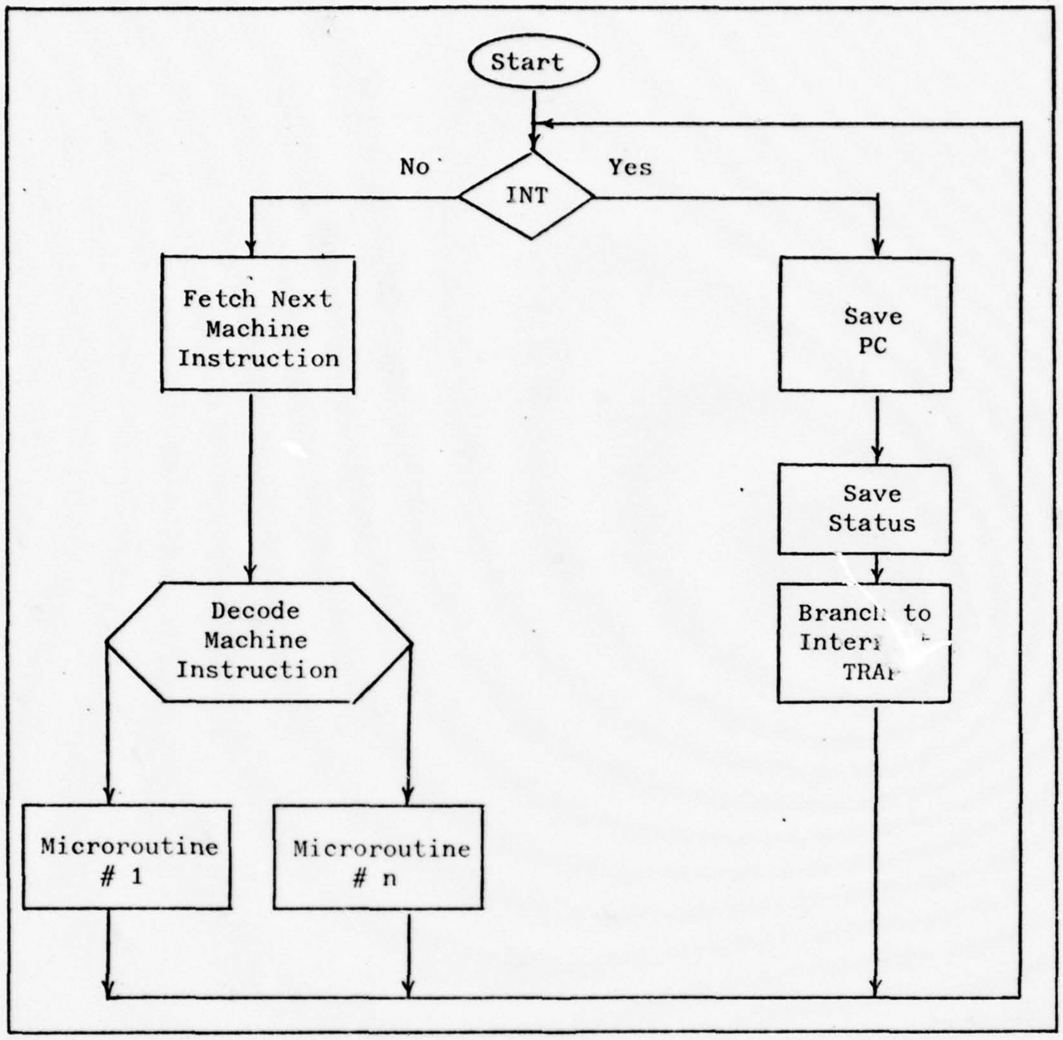


Figure 8. Microprogram Flow

Emulation Example

In order to demonstrate the flexibility of microprogramming and its application to emulation, this section presents the emulation of several sample instructions using the MIME architecture. As discussed in the introduction to this chapter, emulation is considered to be the capability of MIME to execute the instructions of another machine with the same processing results.

Table IX

Sample PDP-11/03 Instructions

Instruction Type	Mnemonic	Instruction	Op Code
Single Operand	COM(B)	Complement Destination	/051DD
Double Operand	MOV(B)	Move Source to Destination	/1SSDD
Signed Conditional Branch	BGE	Branch if ≥ 0	002000
Program Control	RTS	Return from Subroutine	00020R
Trap	TRAP	Trap	104400 - 104777
			/ → 1/0

As a result of the development approach discussed in Chapter I, the authors were familiar with the instruction sets of four minicomputers. Of these, the PDP-11/03 instruction set was considered to be the most flexible and powerful. Therefore, sample instructions for emulation were drawn from it (Ref 8:3.1-5.15). The particular instructions chosen are listed in Table IX. These were chosen in order to illustrate a variety of microprogramming techniques.

The authors used the approach discussed in the previous section to emulate the selected instructions. Therefore, the discussion of this example consists of two parts: High Level Microprogram Flow and Individual Instruction Modules.

High Level Microprogram Flow. The flowchart of Figure 9 shows the microprogram structure used to emulate the PDP-11/03. Each named

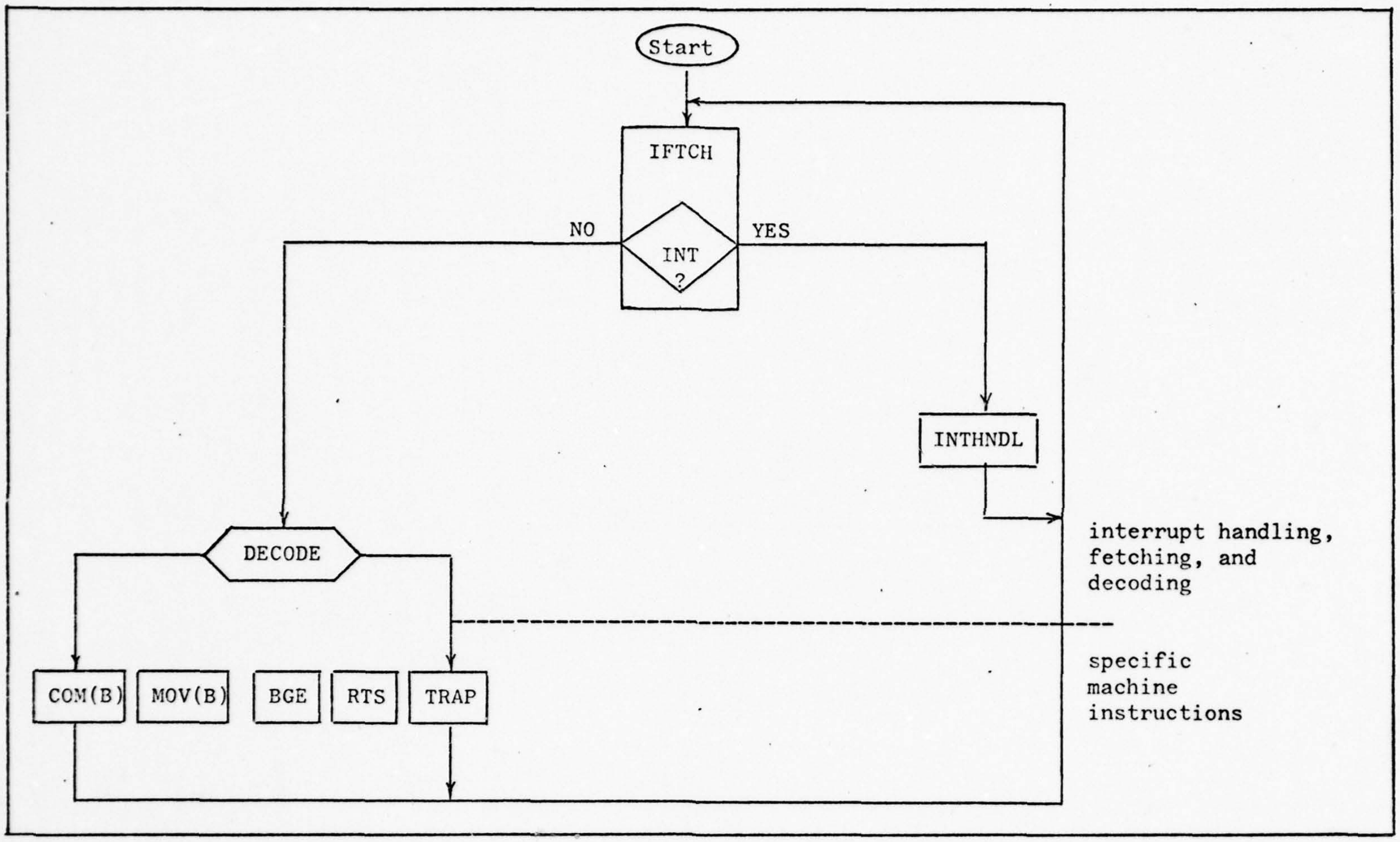

Start
IFTCH
INT ?
NO
YES
INTHNDL
DECODE
COM(B)
MOV(B)
BGE
RTS
TRAP
interrupt handling, fetching, and decoding
specific machine instructions

Figure 9. PDP-11/03 Emulation Structure

block represents a microcode module. Modules above the division line control interrupt handling, fetching, and decoding for all machine instructions. Modules below the line are unique to a particular machine instruction.

Emulation of a particular machine instruction begins with the Instruction Fetch module (IFTCH) which fetches the next instruction from memory if no interrupts are pending. As shown in Figure 10, IFTCH shifts the PC right one place before loading it into the MAR. This is necessary in order to map the PDP-11/03 memory address space into the MIME address space. The corresponding memory word is then fetched from memory to the MBR and subsequently loaded into the IR. The PC is then incremented by two so that it points to the next sequential instruction.

The DECODE module illustrated in Figure 11 controls the decoding of the instruction op code to arrive at the corresponding micromodule starting address. This is accomplished in part by using a PROM to map the op code portion of the IR into starting addresses of other micromodules. If the op code is octal 0002, IR bits 5, 4, and 3 are tested. If IR5,4,3, equals octal 0, a branch to the RTS instruction module is accomplished. Otherwise a branch is made to one of 16 condition code set/clear modules by testing IR bits 3, 2, 1, and 0. If the op code is octal 0000, IR bits 5, 4, and 3 are again tested. In this instance, if IR5,4,3 equals 0, IR bits 2, 1, and 0 are tested to control a branch to one of 8 instruction modules. An invalid instruction has been specified if IR 5,4,3 equals 0. If the op code is other than 0000 or 0002, the mapping PROM output is the starting address of a particular instruction module.

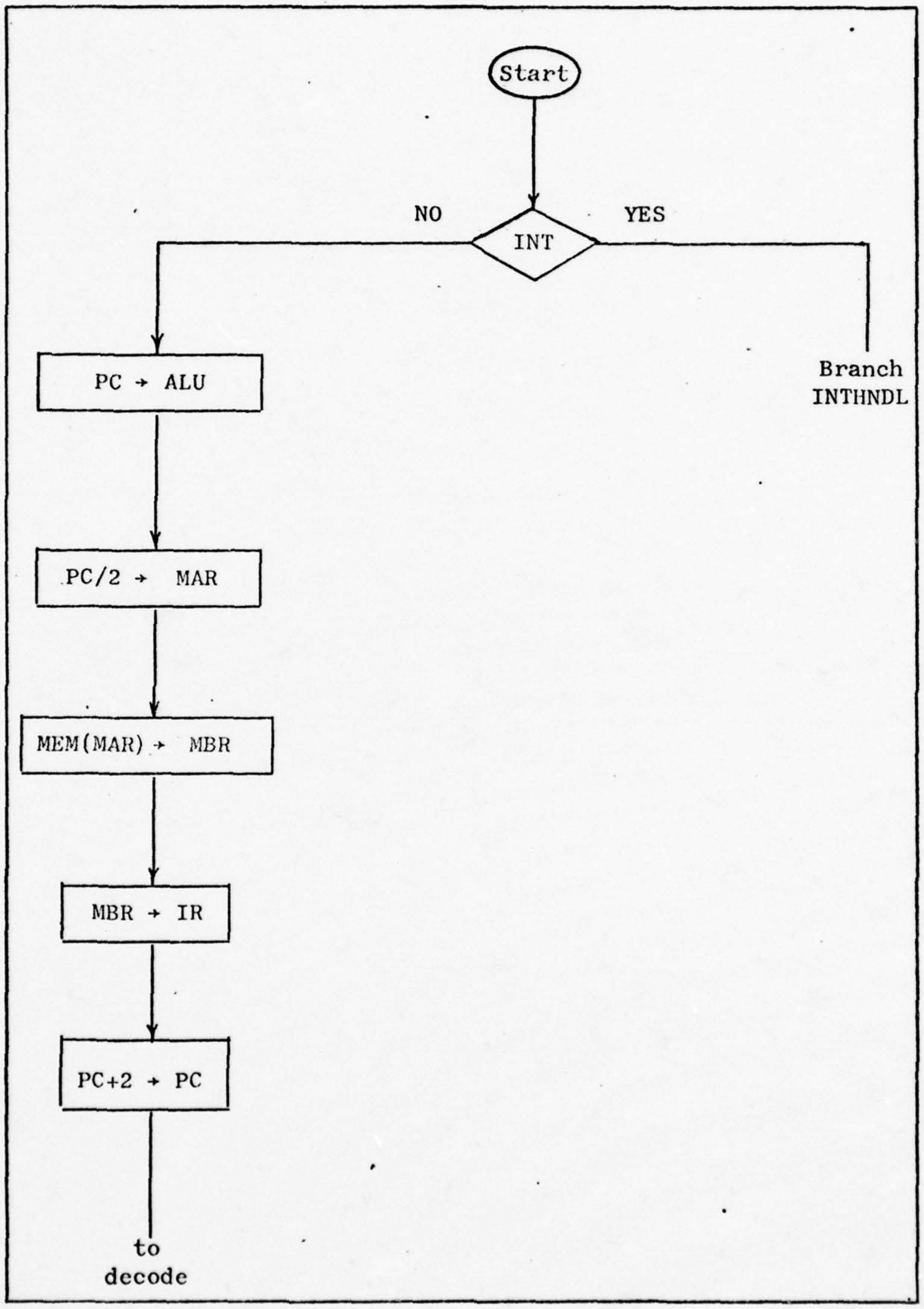


Figure 10. IFTCH Flowchart

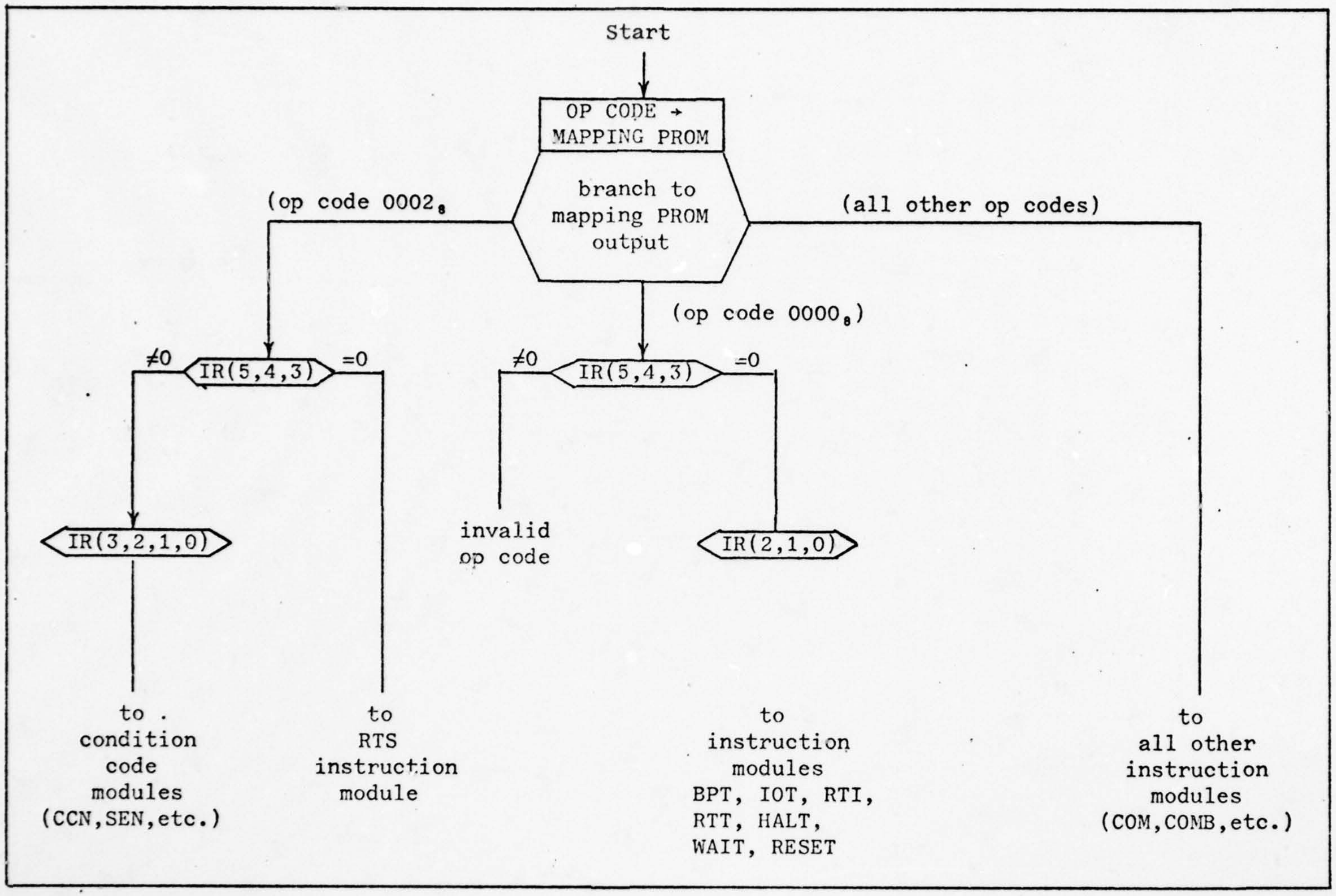

Start
OP CODE → MAPPING PROM
branch to mapping PROM output
(op code 0002₈
(all other op codes)
(op code 0000₈)
≠0
IR(5,4,3)
=0
≠0
IR(5,4,3)
=0
IR(3,2,1,0)
invalid op code
IR(2,1,0)
to . condition code modules (CCN,SEN,etc.)
to RTS instruction module
to instruction modules BPT, IOT, RTI, RTT, HALT, WAIT, RESET
to all other instruction modules (COM,COMB,etc.)

Figure 11. Decode

The Interrupt Handler (INTHNDL) module stacks the current PC and CCR in memory and loads a new PC and CCR from pre-assigned memory locations. The PC then contains the address of the first instruction of the machine language interrupt service routine. Referring to Figure 12, this is accomplished by first calling the micro subroutine PUSH. PUSH decrements R6 (used as the stack pointer) by two, loads the new contents of R6 into the MAR, stores the PC in the memory location addressed by the MAR, and returns. The PC is then stored in the memory addressed by R6. The CCR is similarly pushed onto the stack. The interrupting device is identified by an interrupt vector (IV2,1,0). This vector is used to branch to the particular module necessary to load the MAR with the address of the new PC. The new CCR is in the next sequential location. The PC now points to the machine language interrupt service routine. To execute this instruction, a branch back to IFTCH of Figure 10 is accomplished. After testing for higher priority interrupts, this instruction is executed using the emulation structure discussed in the preceeding paragraphs.

Individual Instruction Modules. The following paragraphs discuss the micro level modules used to implement the register transfers required for each emulated PDP-11/03 instruction of this example.

COM(B) replaces the word (byte) contents of the destination address with its logical complement. Using the decoding algorithm discussed in the previous section, COM and COMB are decoded to separate starting addresses. Thus, they will be discussed as separate modules. As shown in Figure 13, COM begins by calling the DSTDCD micro subroutine to compute and load the MAR with the effective address of the operand (memory reference mode). The reader is referred to the PDP-11/03 Processor Handbook for a comprehensive discussion of instruction formats

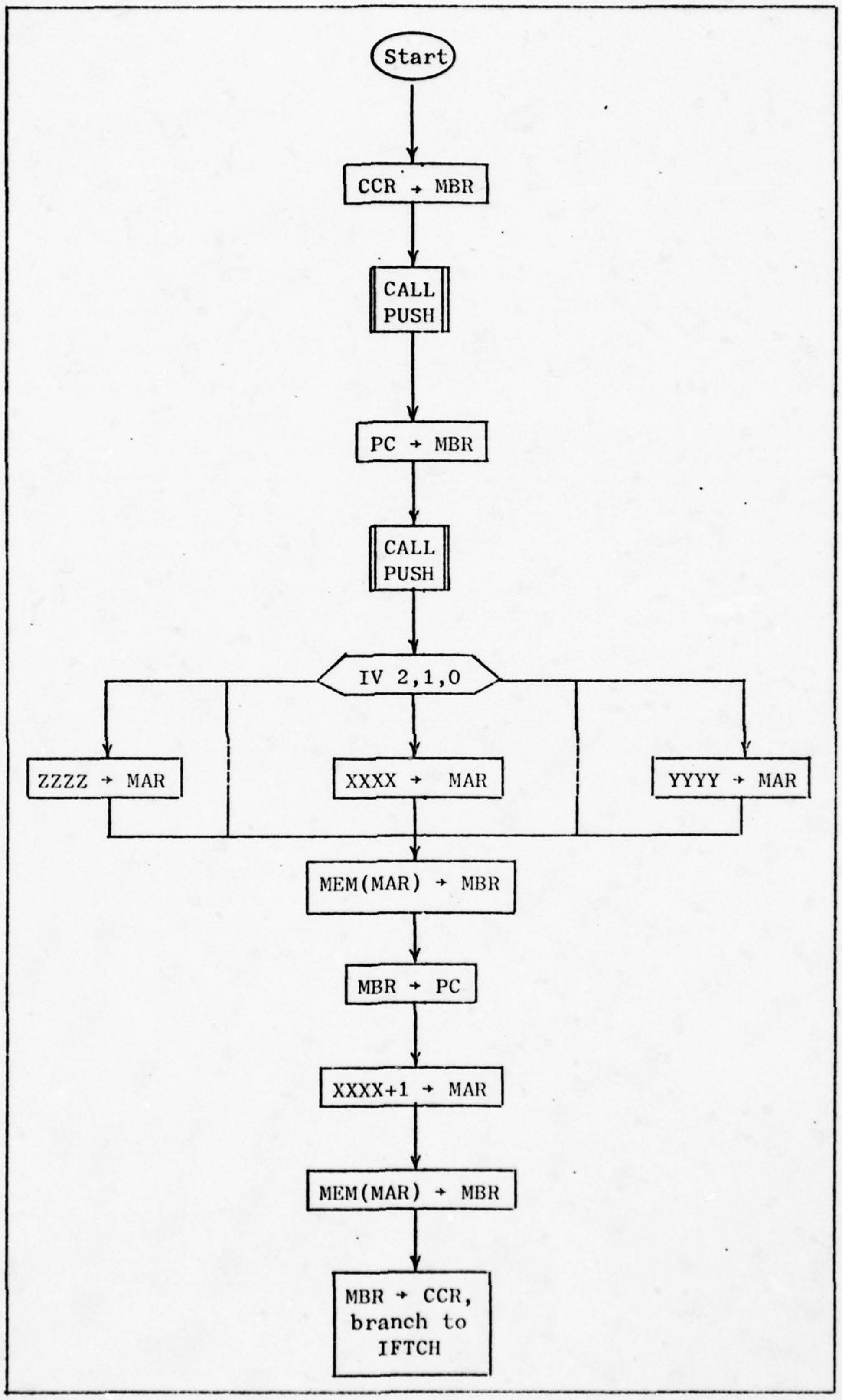


Figure 12. INTHNDL

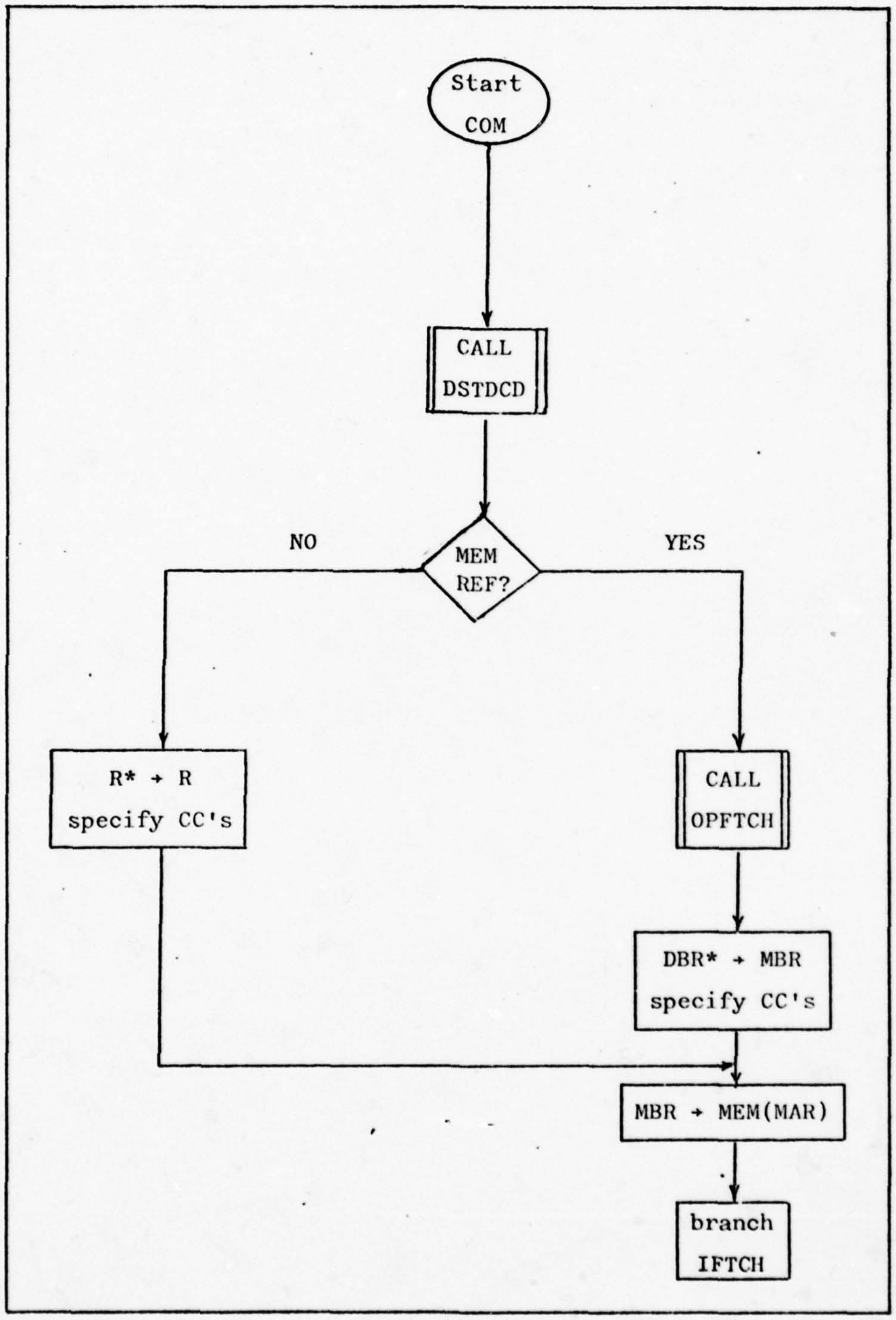


Figure 13. COM Flowchart

and addressing modes (Ref 8:3.1-3.19). If the operand is located in memory, micro subroutine OPFTCH is called to fetch the operand and load it into the ALU Data Buffer Register (DBR). The operand is then complemented and loaded into the MBR. Condition codes specified by COM are generated and stored in the CCR at this time. The MBR is then stored in the initial memory location. This completes execution of memory reference COM so the COM module branches back to the IFTCH. Register reference COM as shown in Figure 13 is similar to the memory reference, but it omits the memory read and write cycles. COMB is also shown in Figure 14. It differs from COM in that only the specified byte is complemented. Register reference implies lower byte, whereas the least significant MAR bit must be tested to find the applicable memory reference byte.

MOV(B) transfers the source operand to the destination location. As shown in Figure 15, MOV calls SRCDCD to find the source operand. OPFTCH is called if memory reference, resulting in the operand being loaded into the ALU DBR. Register reference operands are also loaded into the DBR. The operand is then placed in the Q register and appropriate condition codes generated. DSTDCD is called to determine the destination location, and the operand is stored accordingly. Finally, MOV branches back to IFTCH. As shown in Figure 16, MOVB differs in several respects. First, memory reference sources and destinations require testing to determine the pertinent byte. Second, the source byte must be aligned with the destination byte using SWABQ if they are not both upper or lower. In addition, only a single byte of memory is written into for memory reference destinations. Finally, MOVB requires sign extension through the upper byte for register destinations.

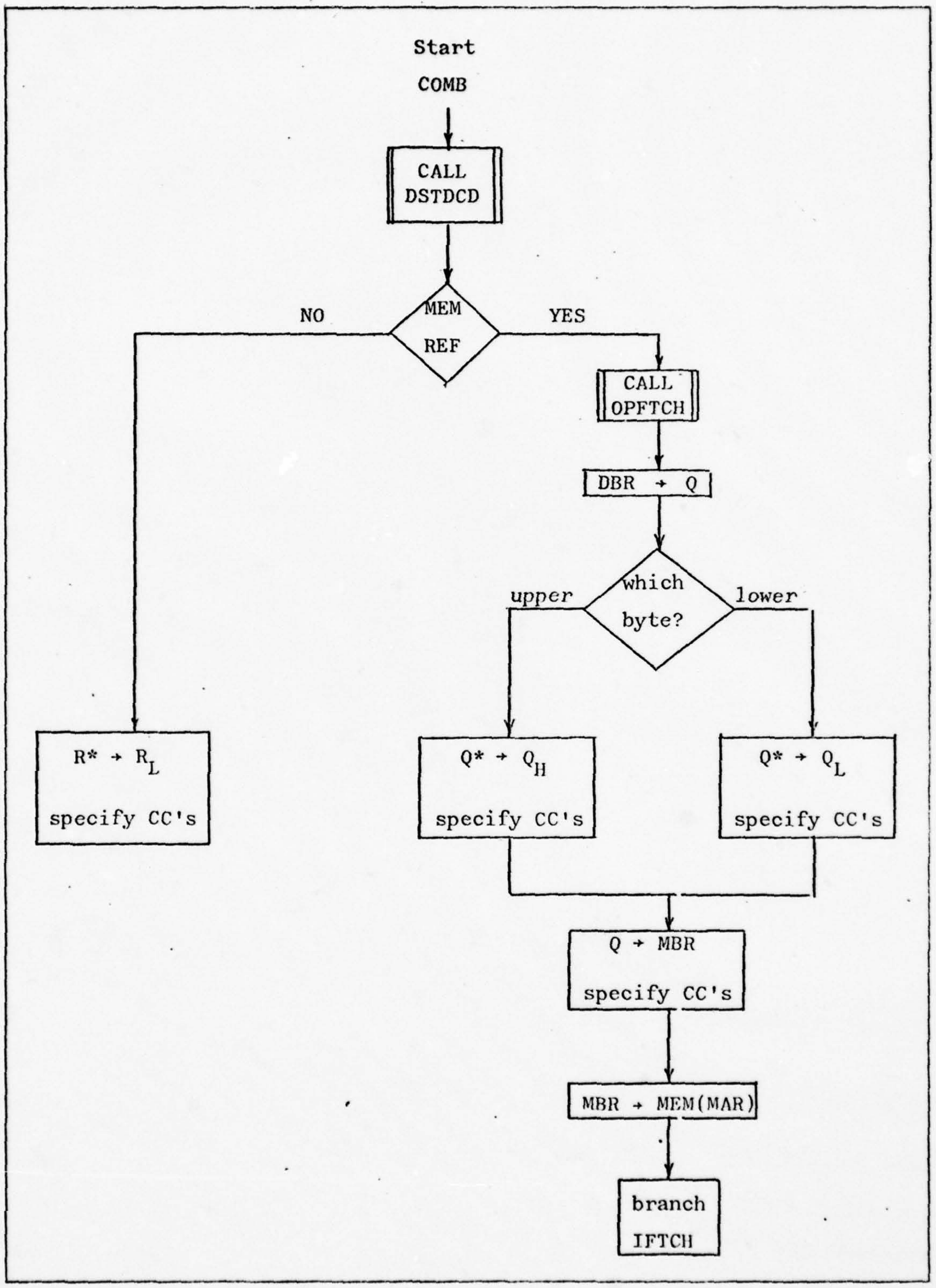


Figure 14. COMB Flowchart

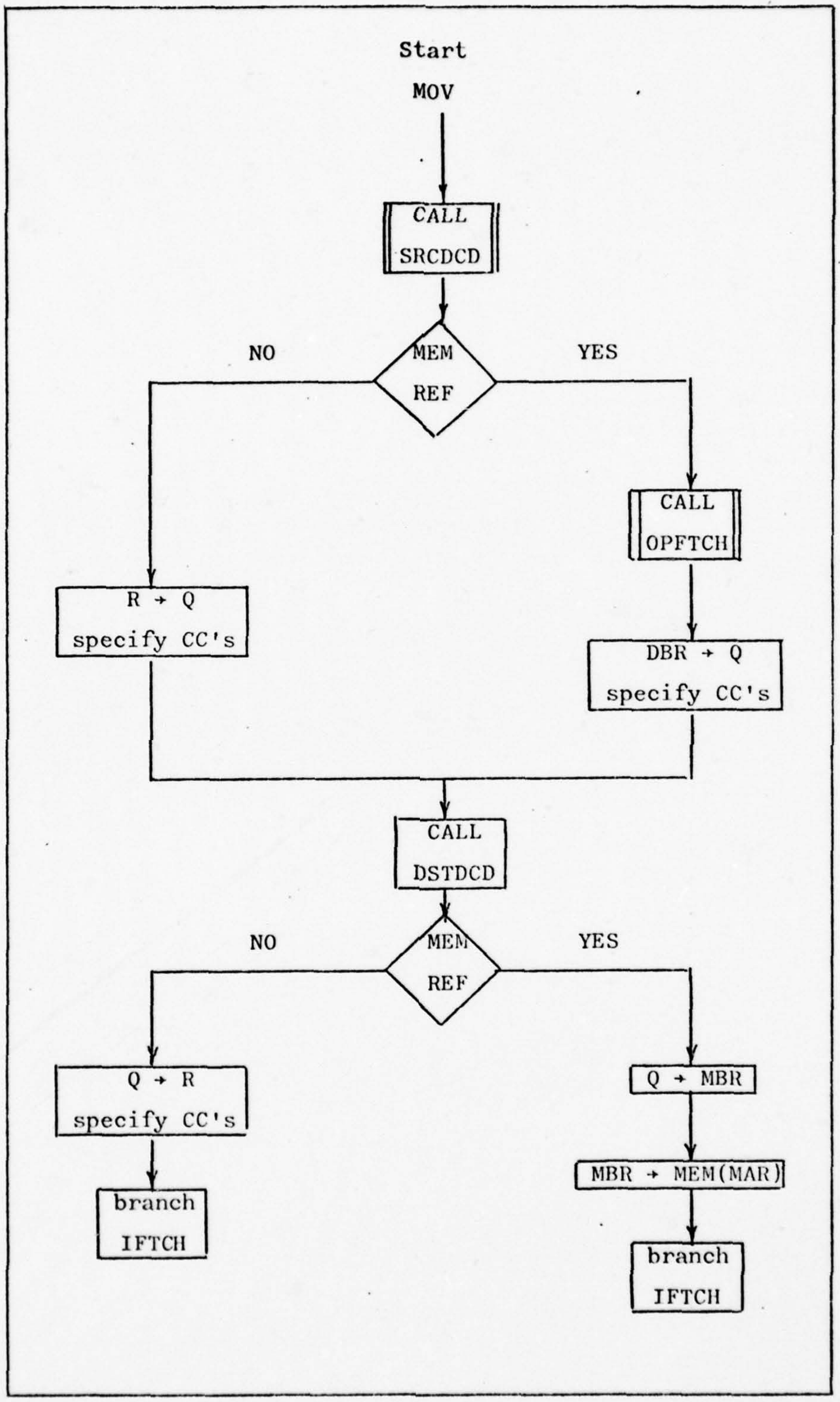


Figure 15. MOV Flowchart

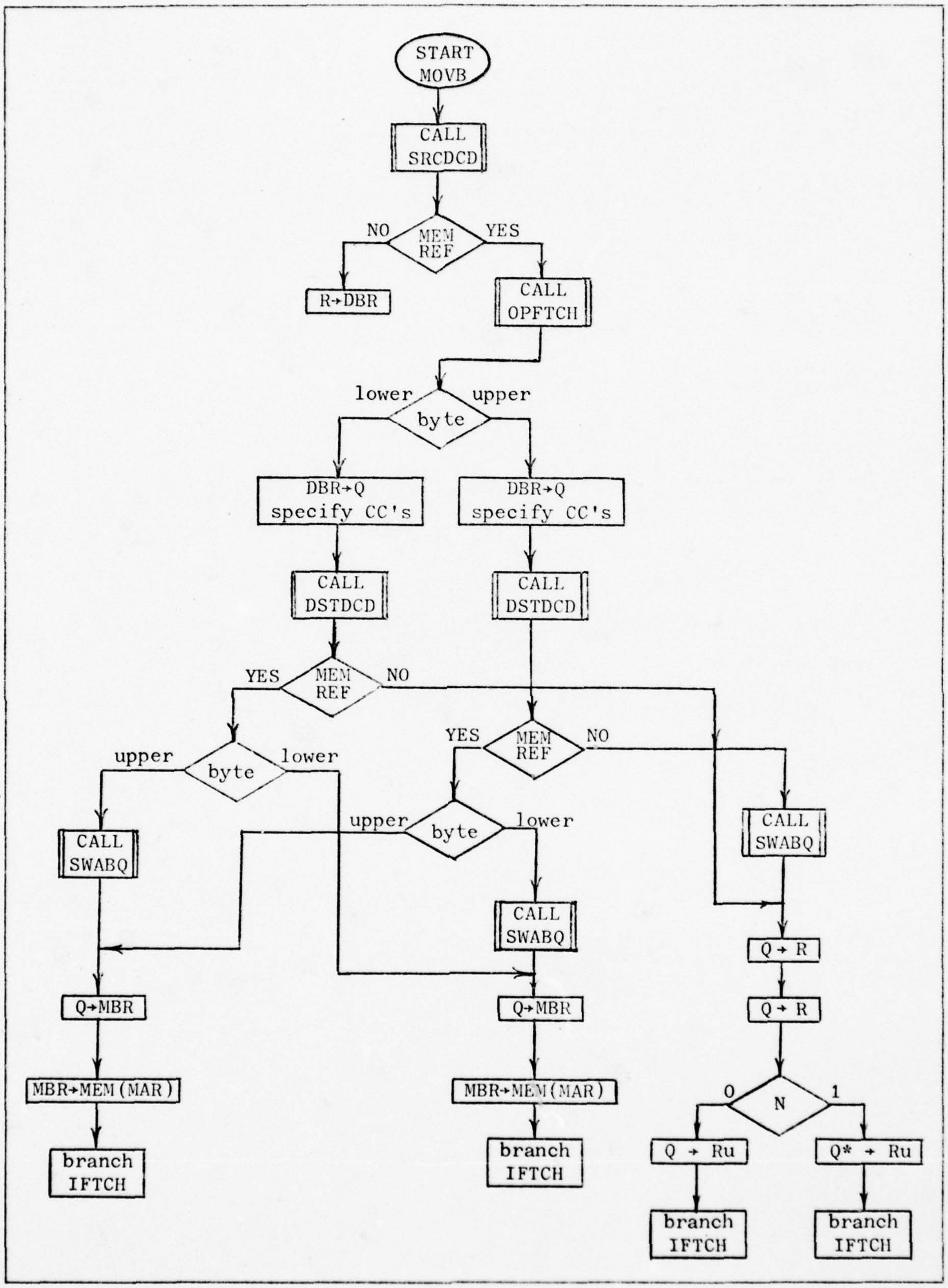


Figure 16. MOVB Flowchart

BGE is a conditional branch if greater than or equal to zero which adds two times the offset (sign extended) to the PC if condition codes N and V are both set or clear. The BGE module, shown in Figure 17, begins by testing condition codes N and V. If they are not both set or clear, the module branches back to IFTCH. Otherwise, the IR is multiplied by 2 and loaded into the lower byte of ALU general purpose register 13 (R13) while the PC is loaded into the DBR. Based on IR bit 7, the lower byte of R13 is sign extended through the upper byte. R13 is then added to the DBR which contains the PC and the result loaded back into the PC. Micro branching back to IFTCH allows the macro branch to occur when the new PC is used to fetch the next instruction, assuming no interrupt.

RTS, return from subroutine, loads the contents of the specified register into the PC, and pops the memory stack into the register. Referring to Figure 18, the RTS module first loads R into the PC and then calls micro subroutine POP to retrieve the top element of the stack. This is then loaded in the register and a branch to IFTCH initiated.

TRAP stacks the current CCR and PC and loads a new CCR and PC from memory locations octal 000036 and 000034 respectively. As shown in Figure 19 this is accomplished by twice loading the MBR and calling PUSH and then loading the PC and CCR with the required constants. As will be discussed further in Appendix A, these constants are stored as part of the microword.

Summary

This chapter has presented a definition of emulation, discussed microprogramming as applied to emulation, and shown an emulation example using PDP-11/03 instructions and the MIME architecture. Appendix A contains details of the example emulation, including representative microcode.

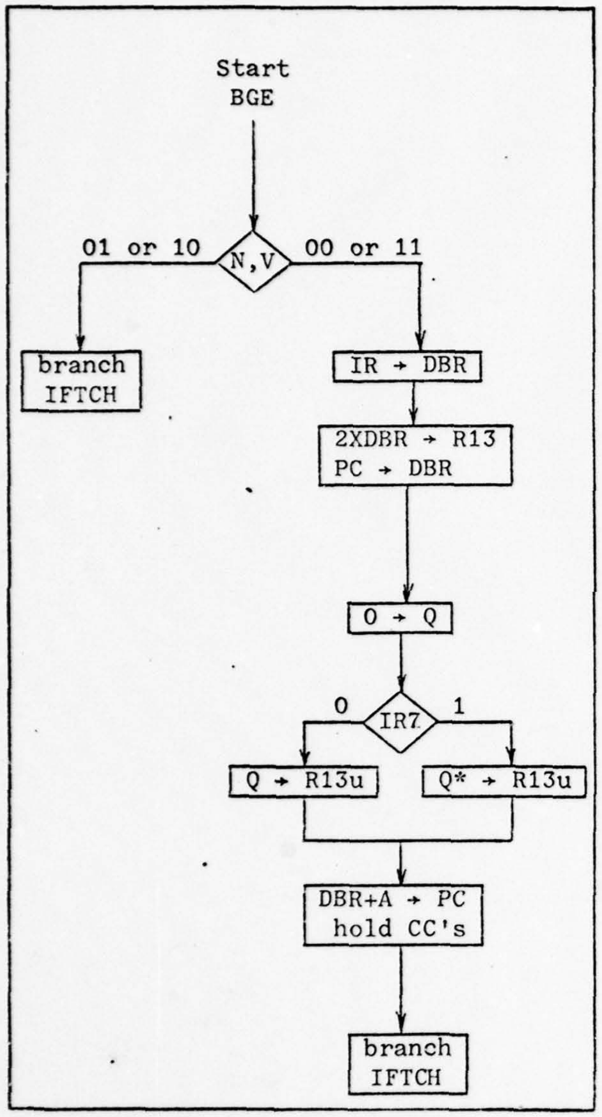


Figure 17. BGE Flowchart

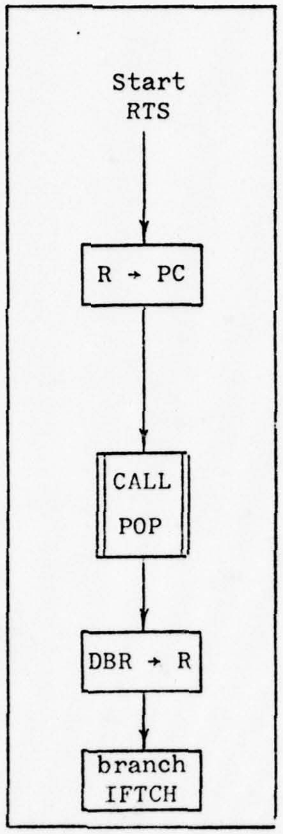


Figure 18. RTS Flowchart

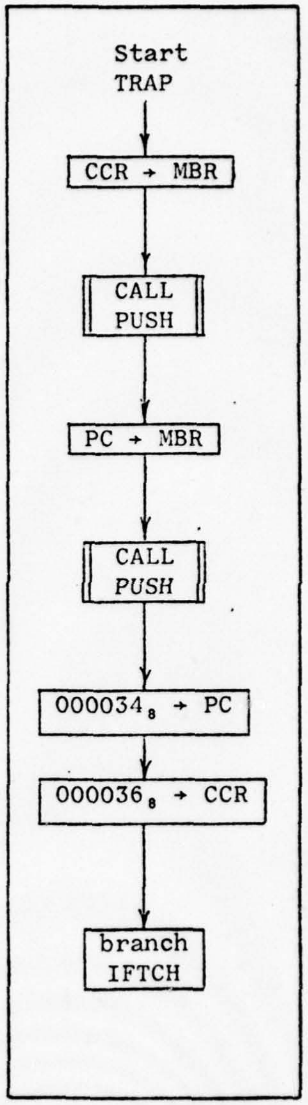


Figure 19. TRAP Flowchart

VI. Results, Recommendations and Conclusions

Results

The purpose of this section is to summarize the significant project results. However, these results were a product of the development approach chosen by the authors. Therefore, before discussing the results, factors that significantly impacted the development will be presented. Results of the hardware realization and emulation example will then be presented.

Development Factors. The literature search which was conducted as the first step yielded no evidence of previous work on a user-microprogrammable general purpose minicomputer specifically designed for educational purposes. It was found that several commercial products were available for use in microprocessor familiarization and microprogramming instruction (Ref 3:18). However, none included the scope of capability envisioned in the problem statement of the thesis. Therefore, the authors attempted to combine the best features of commercial equipment while adding features specifically designed to solve the stated problem.

A second factor of great importance to the development was the schedule estimates. When the schedule was devised, the specified goals were deliberately limited to provide schedule slack for unforeseen difficulties. This provision proved invaluable as a result of the underestimation of the logistics effort required in support of the primary project tasks. The principal components of the logistics effort consisted of selecting and ordering parts and arranging for cabinet fabrication. A probable parts list was compiled early in the requirements definition phase and regularly updated. This allowed difficulties to be identified in time to minimize their impact. The schedule slack originally included allowed time for parts delays and coordination with shop facilities on cabinet fabrication without ultimate impact on the project objectives.

The documentation generated throughout the effort was a third highly significant factor in the accomplishment of the design and fabrication. Three types of documentation were used to maintain configuration control: wiring tables, schematic drawings, and parts list.

The wiring tables presented a complete description of the configuration in terms of component pin numbers. The schematics provided a diagrammatical representation of the electrical connections. A complete parts list was maintained as a key to the designated component numbers.

The sheer volume of the wiring tables would have made their generation nearly impossible and their accuracy uncertain had it not been for the automated procedures described in Appendix B. The ready availability of this data facilitated corrections and modifications and minimized the problems encountered during integration and checkout. The FP module, which consists of two circuit cards, was a specific example. One card was designed and constructed before the configuration management procedures were fully implemented, but the design and fabrication of the other was fully controlled. During checkout of the FP wiring problems were encountered only in the first card.

Hardware. Chapters II and III presented the requirements and Chapter IV provided significant details concerning the hardware implementation. Complete wiring tables, schematics, and parts list are included in Appendix A. Also included is the detailed test procedure which was devised to verify that MIME had met the requirements and was fully operational. The two basic test approaches used are described below.

The FP module was checked using test instruments to verify correct circuit operation. This consisted of either comparing a signal level or a waveform present at a particular point to that prescribed by the test procedures. The test order was designed so that each succeeding test was built on those previously successful. An unsuccessful test suspended the procedure until the fault was corrected.

After the FP was certified operational it was used to conduct the test procedures specified for the remaining MIME modules. These consisted of using the front panel capability to load or display various quantities and comparing the result to that specified by the test procedures. When a discrepancy occurred, the procedure was suspended and the fault isolated using the troubleshooting techniques and functional descriptions contained in the MIME User's Manual, Appendix A.

The successful completion of the test procedures demonstrated that the hardware was operational and that it met the specified requirements. These tests were conducted as the final step of the hardware checkout and the results recorded in a test log.

Emulation. One facet of the thesis goal specified in Chapter I was to provide the microcode necessary to demonstrate a representative sample of MIME's capabilities. The application of the emulation approach described in Chapter V required microcode to be generated in two areas:

for the implementation of the emulation structure discussed in Chapter V and for implementation of specific sample macroinstructions. For reasons discussed in Chapter V, PDP-11/03 instructions were used as test macroinstructions.

Test procedures consisted of entering the microcode into the CS and then attempting to execute the emulated instructions. After the execution of each macroinstruction the contents of the appropriate registers and memory locations were compared against the correct values specified by the procedure. The test results were recorded in the test log and the successful completion of the test demonstrated that this facet of the goal had been accomplished.

Recommendations

The recommendations which resulted from this project fall into two categories: enhancements and applications. MIME has the potential for expansion and increased cpability with only relatively minor additions or modifications, and the authors recommend several. MIME was specifically created as an educational tool so the authors have recommended several applications where they feel MIME might best be utilized.

Enhancements. The enhancements recommended for MIME are in the following categories: Control Store, Memory, Arithmetic and Logic Unit, I/O, and Emulation.

Although only 1K of CS EPROM was initially provided, MIME had the capacity for a total of 4K words of CS. It is recommended that the CS be expanded to full capacity in order to allow more complex or multiple emulations to be resident in MIME. Multiple resident emulations would also require multiple mapping PROMS or physical substitution for each different emulation.

Substitution of Read-Only-Memory (ROM) for EPROM would increase microinstruction execution speed by a factor of two when operated in the asynchronous mode with a faster clock. It is recommended that the feasibility of doing so be investigated thoroughly before implementation. The increased speed could improve emulation performance or merely provide a greater potential for waiting.

MIME memory may also be expanded from the initially provided 1K to a maximum of 64K. Although a need for the total capacity is unlikely, it is recommended that memory be expanded to a minimum of 8K. This is the minimum that would be required to provide the capability of using operating system software written for the emulated machine (Ref 19: Sect. 2,1-1).

At the time parts were being selected for MIME, Advanced Micro Devices had just released data on a new microprocessor, the Am2903 (Ref 20). This chip represented a substantial increase in capability over the Am2901. For example, it includes hardware multiply and divide and provision for an expandable register file. However, its anticipated availability fell too late in the project schedule to allow it to be used. It is recommended that the feasibility of replacing the Am2901's in MIME with Am2903's be investigated.

Although an RS 232 interface has been provided (Ref 21), MIME has the capacity to handle additional peripherals. It is recommended that an interface for a mass storage unit such as a cassette be provided. This would expand MIME processing capability as well as provide a second example of an I/O interface.

As addressed earlier, only representative PDP-11/03 instructions were emulated and verified. In order to provide MIME with a complete resident assembly language instruction set, it is recommended the emulation be completed and stored in the CS EPROM.

Applications. The authors feel that MIME will have greatest applications as the subject of student projects in support of the AFIT digital engineering/computer science curriculum. It is recommended that MIME and the system documentation be used in conjunction with lecture and structured laboratory instruction. MIME was designed for educational purposes, but the system documentation was written assuming that users will have a basic understanding of the principles of computer operation and architecture. Also, the system documentation does not include learning goals or experimental procedures. However, once they have been defined, the documentation does provide the user with all the information necessary for employing MIME to achieve his objectives. Therefore, in order for the user to obtain the greatest benefit, it is important that theoretical background and laboratory guidance be provided. It is anticipated that MIME will prove most useful in the following areas: support of course objectives in EE 6.87 (Ref 22:60) both directly and indirectly. It is an example of a general purpose minicomputer constructed using current LSI technology, and it can be used to demonstrate techniques of computer control and microprogramming applied to I/O processing. The analysis required to generate microcode for handling programmed, interrupt, and DMA type I/O transfers will contribute to the user's understanding and exploration of various I/O techniques.

MIME can be used for general purpose instruction in microprogramming applied to the many facets of computer control. An analysis of the microcode required to implement a given computer operation should yield an understanding of the required data transfers, control signals, and timing.

This discussion is also applicable to investigation of emulation with the addition of a specified instruction set as a target.

Conclusion

The scope of this project included activities that on a small scale closely paralleled the Air Force development cycle (e.g. requirements definition, conceptual phase, design/cost tradeoffs, development, fabrication, test and integration, and turnover and transition). Because such a varied range of activities, encompassing both theory and practical application, was required, the benefit derived was much greater than anticipated.

MIME has been used as the subject of projects in support of two courses, EE 6.87 and EE 6.88 (Ref 22:60). The EE 6.87 project consisted of the design of hardware and microprograms necessary to implement an RS232 interface (Ref 21). This project allowed students to apply microprogramming as a means of controlling I/O transfers. The EE 6.88 project utilized MIME to familiarize students with the concepts and applications of microprogramming. This project consisted of having students emulate various minicomputers and microprocessors using the MIME architecture.

The degree to which MIME possesses a generalized architecture and is educationally configured is a subjective evaluation ultimately left to the reader. However, that MIME has utility for graduate level investigations in the areas of computer control and microprogramming was demonstrated by the projects discussed above. As a result, the authors have concluded that the objective of the investigation as stated in Chapter I has been accomplished.

Bibliography

1. Agrawala, Ashok K. and Tomlinson G. Rauscher. Foundations of Microprogramming. New York: Academic Press, Inc., 1976.

2. Brick, Jim and John Mick. "Microprogramming Ups Your Options in μP-System Design", EDN, 23: 105-110 (January 1978).

3. Am2900 Learning and Evaluation Kit User's Manual. Sunnyvale: Advanced Micro Devices, 1976.

4. Am2900 Bipolar Microprocessor Family. Product Brochure. Sunnyvale: Advanced Micro Devices, 1976.

5. A Microprogrammed 16-Bit Computer. Applications Note. Sunnyvale: Advanced Micro Devices, 1976.

6. Abd-alla, Abd-elfattah M. and Arnold C. Meltzer. Principles of Digital Computer Design. Englewood Cliffs: Prentice-Hall, Inc., 1976.

7. Weitzman, Cay. Minicomputer Systems. Englewood Cliffs: Prentice-Hall, Inc., 1974.

8. PDP-11/03 Processor Handbook. Maynard: Digital Equipment Corporation, 1975.

9. How to Use the Nova Computers. Southboro: Data General, 1972.

10. 21MX Computer Series Reference Manual. Cupertino: Hewlett Packard Company, 1974.

11. Madnick, Stuart E. and John J. Donovan. Operating Systems. New York: McGraw-Hill, 1974.

12. Mick, John R. and Jim Brick. Microprogramming Handbook and Am2910 Emulation (Second Edition). Sunnyvale: Advanced Micro Devices, 1977.

13. Advanced Micro Devices MOS/LSI Data Book. Sunnyvale: Advanced Micro Devices, 1976.

14. Baker, Charles A. and Walter F. Grether. "Displays", Human Engineering Guide to Equipment Design, Edited by Harold P. Van Cott and Robert G. Kinkade. Washington, D. C.: U. S. Government Printing Office, 1972.

15. Abrams, Marshall D. and Phillip G. Stein. Computer Hardware and Software. Reading: Addison-Wesley Publishing Company, 1973.

16. Husson, Samir S. Microprogramming: Principles and Practices. Englewood Cliffs: Prentice Hall, Inc., 1970.

17. "Computers: Consolidating Gains", Spectrum, 15: 26-30 (January 1978).

18. Texas Instruments Microprocessor Learning System. Bulletin CB 184-A. Dallas: Texas Instruments Inc., 1976.

19. PDP-11 Software Handbook. Maynard: Digital Equipment Corporation, 1975.

20. Coleman, Vernon, Michael W. Economidis, and William J. Harmon Jr. The Next Generation Four-Bit Bipolar Microprocessor Slice - the Am2903. Unpublished Technical Data. Sunnyvale: Advanced Micro Devices.

21. Pennett, Gary, and Saleem Iftekhar. Design of an RS232 Interface for the Microprogrammable Minicomputer Emulator. Unpublished laboratory report. Wright-Patterson Air Force Base: Air Force Institute of Technology, 1978.

22. Air Force Institute of Technology Catalog. Wright-Patterson Air Force Base: United States Air Force, 1976.

23. ASD Computer Center CDC NOS/BE User's Guide. Revision D. Wright-Patterson Air Force Base: Aeronautical Systems Division, 1976.

24. Purvis, Richard E. and Ronald D. Yoho. Use of Fortran IV in Computer-Aided Wiring Documentation Generation. Unpublished Special Study Report. Wright-Patterson Air Force Base: Air Force Institute of Technology, 1978.

Appendix A

MIME User's Manual

Preface

MIME (Microprogrammable Minicomputer Emulator) is an educationally oriented, user microprogrammable, general purpose minicomputer. It was designed and constructed as a master's thesis project at the Air Force Institute of Technology specifically for use in laboratory instruction and student design projects in the areas of computer control, microprogramming, and emulation (Ref 1).

This manual has several purposes: describe MIME characteristics, provide the information necessary for effective utilization of MIME, and facilitate maintenance and modifications. Section I, System Description, introduces the architecture and system hardware, summarizes the possible information transfers, and concludes with detailed description of each functional module. Sections II and III provide operating procedures and microprogramming techniques respectively. Section IV concludes with maintenance information. A complete list of terms and abbreviations is included on page 77. The authors believe that MIME will provide valuable hands-on experience that will greatly contribute to the student's understanding of computer control and microprogramming.

Contents

List of Tables

List of Figures

List of Terms

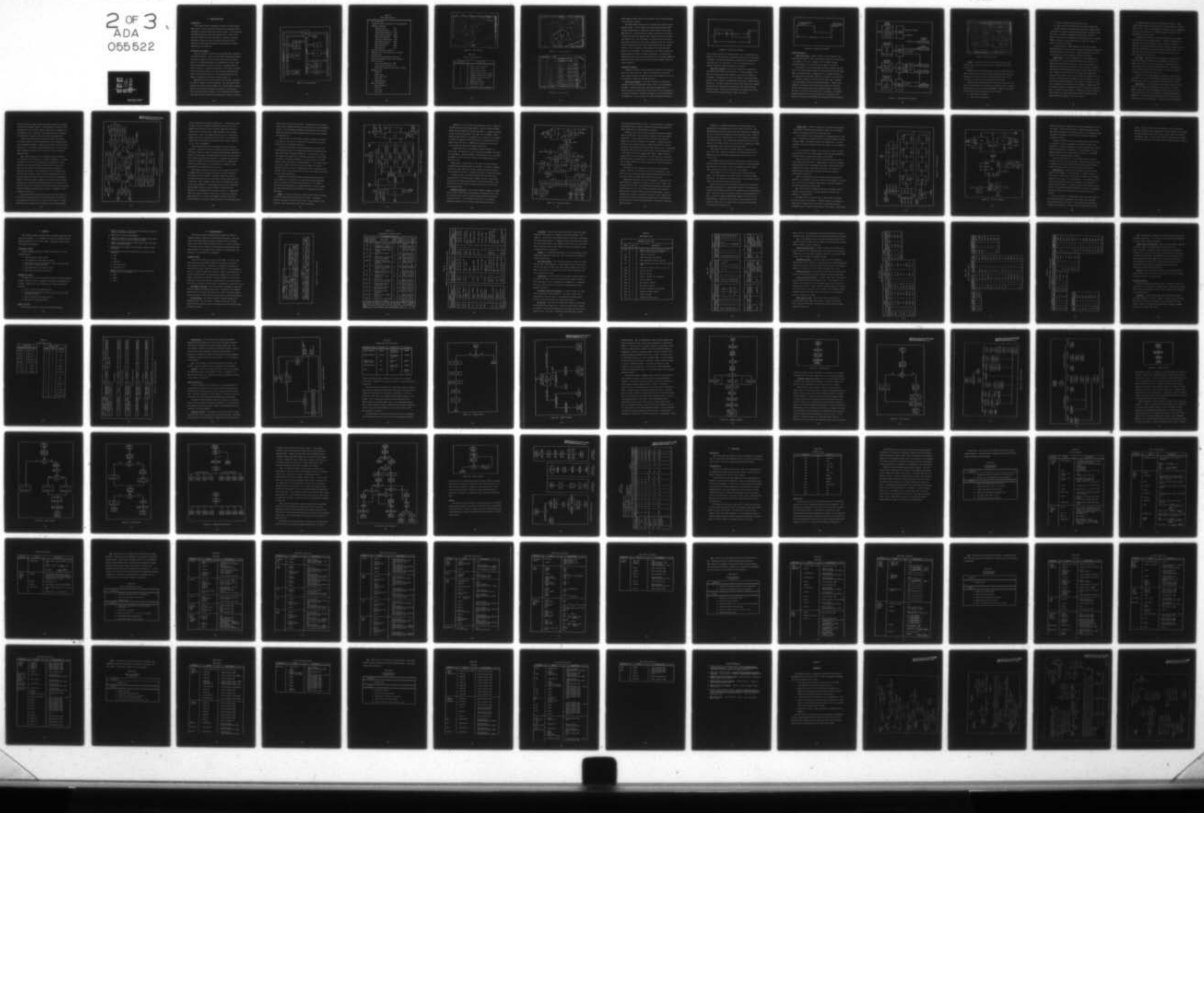
AB	Address Bus
ALAT	A address latch
ALB	A less than B (Comparison of bytes A & B)
ALU	Arithmetic Logic Unit
AMD	Advanced Micro Devices
AMUX	A address mux
backplane	Wiring between edge connectors
BAR	Base Address Register
BLA	B less than A (Comparison of bytes A & B)
BLAT	B address latch
BMUX	B address mux
BP	Breakpoint
BRMUX	Branch mux
CCR	Condition Code Register (macro level)
CCU	Computer Control Unit
CS	Control Store
CSB	Control Store Buffer
DB	Data Bus
DBR	Data Buffer Register: buffers DB to D inputs of Am2901's
DC	Display code
DMA	Direct Memory Access
EPROM	Electrically programmable read-only memory
FP	Front Panel
hex	Hexadecimal
i	Subscript used to indicate integer
IMUX	Instruction mux: chooses input to 2901's

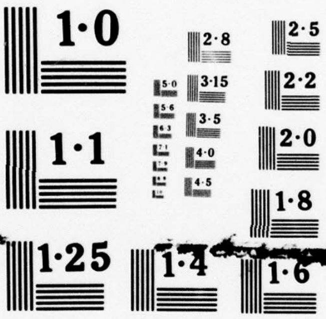
I/O	Input/Output module
IR	Instruction Register (CCU)
KYBRD	Keyboard
L.S.	Least Significant
MAB	Microaddress Bus
macro	Designation of the machine language level of operation
MAR	Memory Address Register (MEM1)
MB	Memory Bus
MBR	Memory Buffer Register (ALU)
MCCR	Micro Condition Code Register (ALU)
MDB	Microdata Bus
MEM1	Memory module: main program storage
MEMFF	Memory flip-flop: specifies read or write
micro	The microprograms level of operation
microword	Group of bits used to generate control signals
MIME	Microprogrammable Minicomputer Emulator
MR	Mask Register (in Am2914)
MLC	Micro Loop Counter (CCU)
MPMUX	Mapping mux
MPROM	Mapping PROM
MSKB	Mask Buffer: buffers PL 63-56 to DB
M.S.	Most Significant
MSP	Miscellaneous Signal Panel
MTCM	Micro Test Condition Mux (ALU)
mux	Multiplexer
N.A.	Not Applicable
PC	Program Counter
PL	Pipeline Register (CCU)

POLMUX	Polarity Mux (CCU)
Q	Q register in Am2901's
RAM	Random Access Memory
Ri	One of 16 2-port RAM locations in Am2901's
DFF	Direction Flip-flop (I/O)
SR	Status Register (Am2914)
TCM	Test Condition Mux (ALU)
TCRB	Test Condition Register Buffer (ALU)
WCR	Word Count Register (I/O)
YBFR	y Buffer: buffers output of Am2901's to DB

AD-A055 522 · AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 9/2
MIME: MICROPROGRAMMABLE MINICOMPUTER EMULATOR. (U)
MAR 78 R E PURVIS, R D YOHO
UNCLASSIFIED AFIT/GE/EE/78-6 NL

2 OF 3
ADA
055522





NATIONAL BUREAU OF STANDARDS
MICROCOPY RESOLUTION TEST CHART

I. System Description

Introduction

This section presents information necessary for understanding MIME operation, and it provides the basis for the operating, microprogramming, and maintenance sections which follow. The Architecture and Hardware subsection addresses MIME structure in terms of the functional modules and their organization; Information Transfers summarizes the data flow that is possible utilizing the MIME architecture; and Module Descriptions presents details of module implementation and operation.

Architecture and Hardware

Figure A1 contains a functional block diagram which illustrates the MIME architecture. MIME is a bus oriented, functionally modularized digital computer having the general characteristics listed in Table AI. The functional modules are implemented on circuit cards having the designations listed in Table AII. The MIME hardware is shown in Figure A2. Except for Front Panel 1 (FP1) which is affixed to the cabinet front panel, all circuit cards are rack mounted as shown in Figure A3 and Figure A4. Terms and acronyms used in the manual are summarized in the list of terms on page 77.

MIME makes use of a 16 bit data word, Data Bus (DB), and Address Bus (AB) for register and memory transfers. The front panel module provides the operator with the capability to control MIME's operating mode as well as load and display macro (machine) level and micro level register and memory contents. The ALU performs the arithmetic/logic operations and contains the general purpose registers, the Data

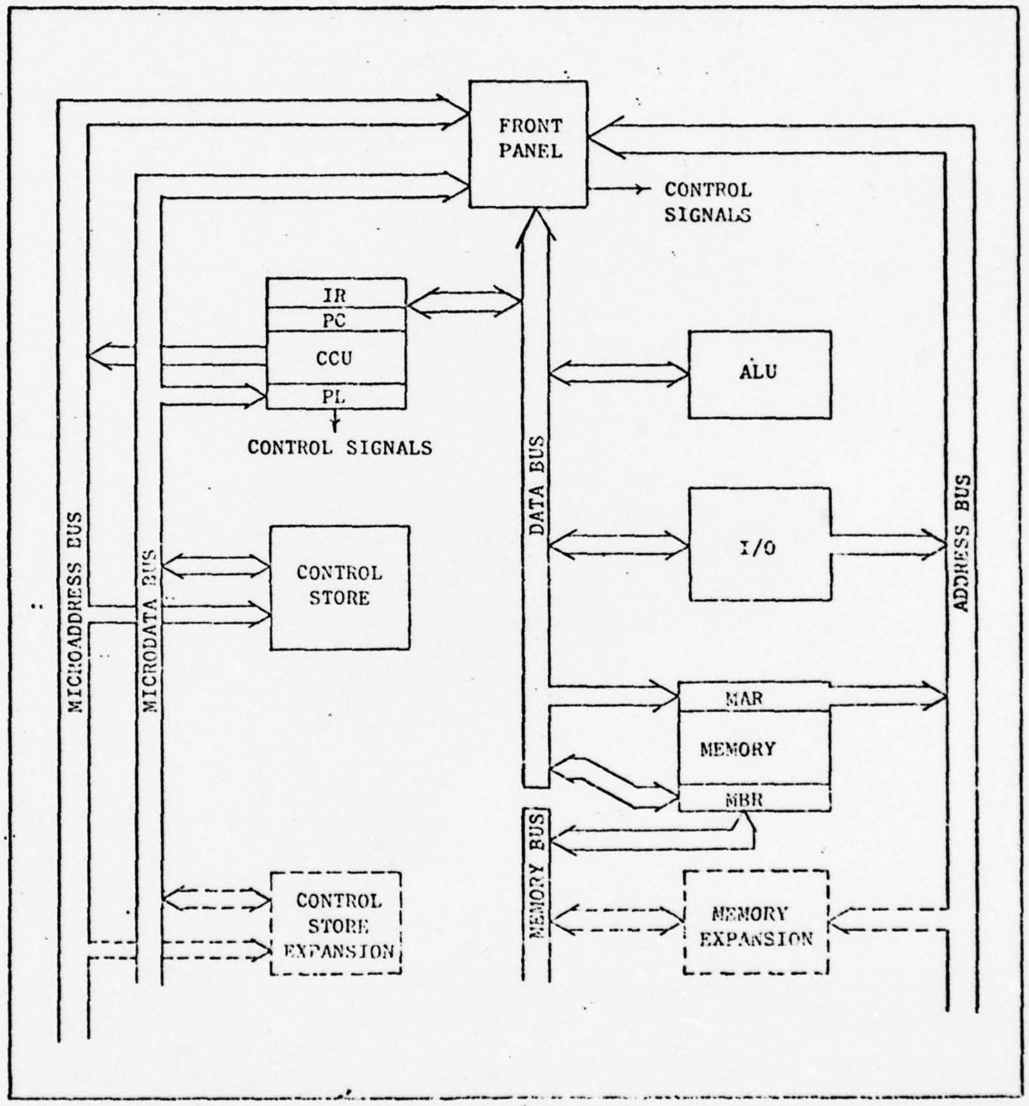


Figure A1. MIME Architecture

Table AI

MIME Characteristics

16 bit data word	
17 general purpose registers (Q, R0 through R15)	
Dedicated registers	
Program Counter	(PC)
Instruction Register	(IR)
Memory Address Register	(MAR)
Memory Buffer Register	(MBR)
Data Buffer Register	(DBR)
Condition Code Register	(CCR)
Base Address Register	(BAR)
Word Count Register	(WCR)
Status Register	(SR)
Mask Register	(MR)
64K words main memory capacity	
Eight levels of vectored priority interrupt	
1.43 MHz clock	
Synchronous memory reference operations	
4K microword maximum Control Store capacity	
64 bit microword	
12 bit microaddress bus (MAB)	
64 bit pipeline register (PL)	
8 bit micro condition code register (MCCR)	
Operating modes	
automatic	
single step	
auto single step	
micro step	
auto micro step	
Power requirements	
110V, 50-60Hz, 2A	
21A @ 5V	
1.6V @ -5V	
2.4 @ 12V	

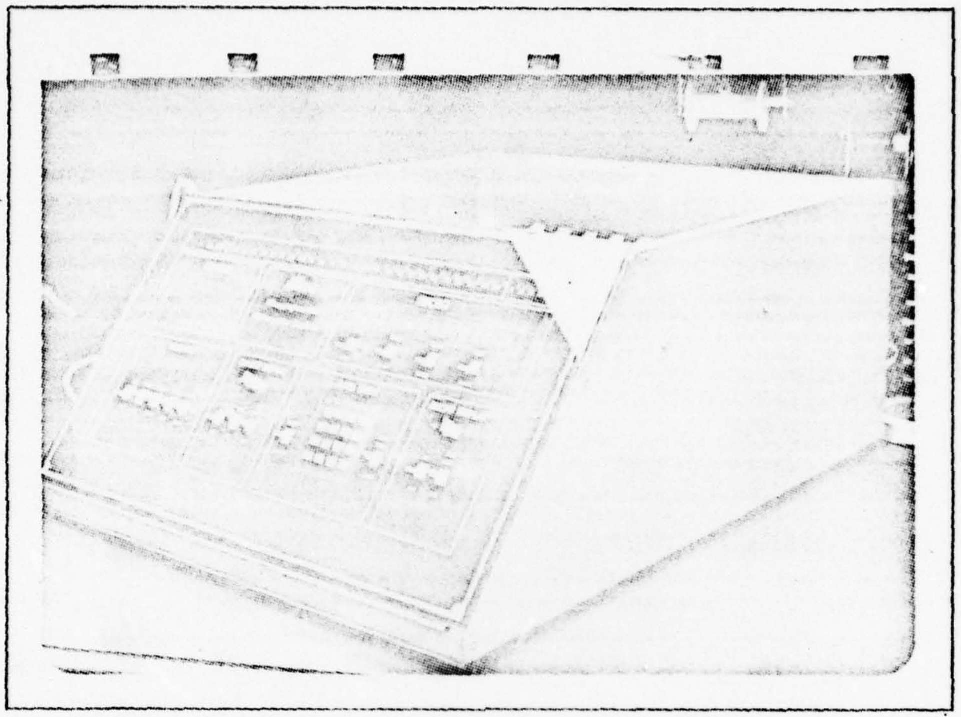


Figure A2. MIME Hardware

Table AII

Circuit Card Identifiers

Identifier	Circuit Card
A	Control Store 2 (CS2)*
B	Front Panel 2 (FP2)
C	Control Store 1 (CS1)
D	Computer Control Unit 1 (CCU1)
E	Computer Control Unit 1 (CCU2)
F	Arithmetic Logic Unit (ALU)
G	Memory 1 (MEM 1)
H	Input/Output (I/O)
P	Front Panel 1 (FP1)
	* Reserved for future expansion

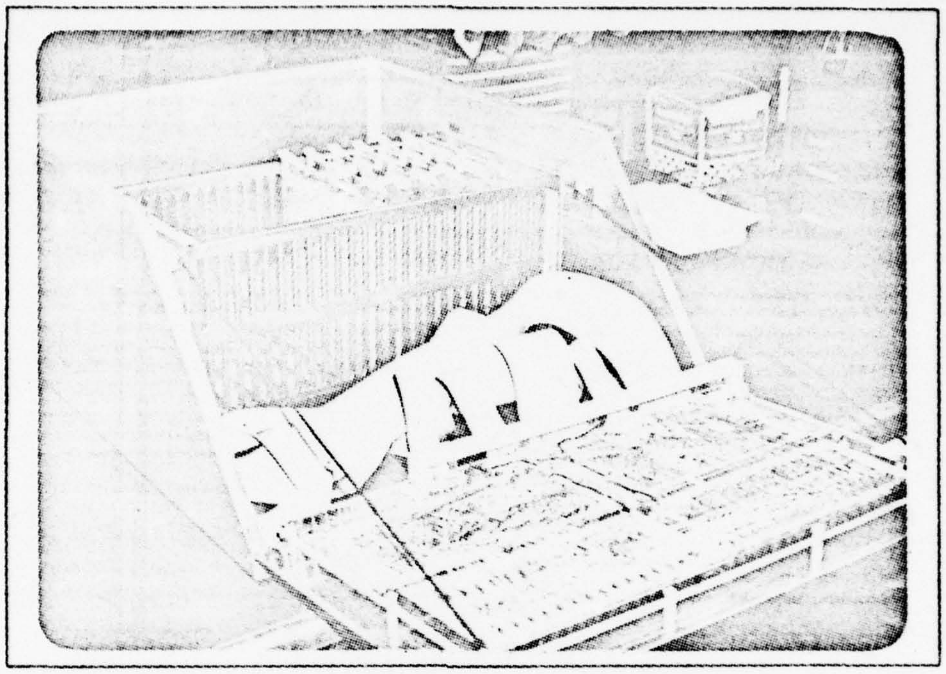


Figure A3. Card Rack Mounting

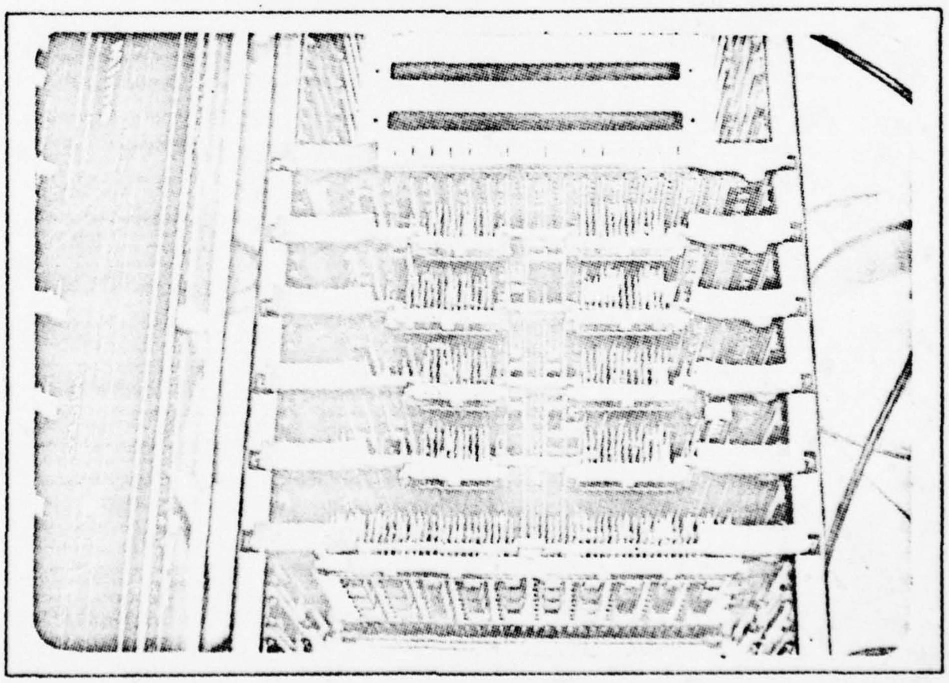


Figure A4. Card Positions

Buffer Register (DBR), Condition Code Register (CCR), and Microinstruction Code Register (MCCR).

The memory module (MEM1) provides a random access memory for main program storage. It also contains the Memory Address Register (MAR) and Memory Buffer Register (MBR) which provide buffers to the buses. MIME operation is controlled by the Computer Control Unit (CCU) which contains the Instruction Register (IR), Program Counter (PC) and Pipeline Register (PL). The CCU decodes the instruction which came from the location specified by the PC and is contained in the IR. Execution is then accomplished by the control signals generated by the contents of the PL. The PL contains a microword which was fetched from the Control Store (CS) module. The Input/Output (I/O) module provides the means necessary for MIME to interface peripherals. It contains the hardware necessary to provide the vectored priority interrupt capability and the Word Count Register (WCR) and Base Address Register (BAR) used in DMA transfers.

Information Transfers

Information transfers within MIME may be divided into two classes: micro level and macro level. Micro level transfers are defined as those between CS and CCU and macro level as those between any of the 16 bit registers of MIME (e.g. IR, MBR, R0, etc.).

Micro Level Transfers. Micro level transfers are essentially successive fetches of microwords from CS to the PL via the microdata Bus (MDB). Using the first level pipelining technique (Ref 2), a single microword is fetched from CS concurrently with the execution of the previous microword. Referring to Figure A5, the current microword is clocked into the PL at point A, the clock low-to-high transition,

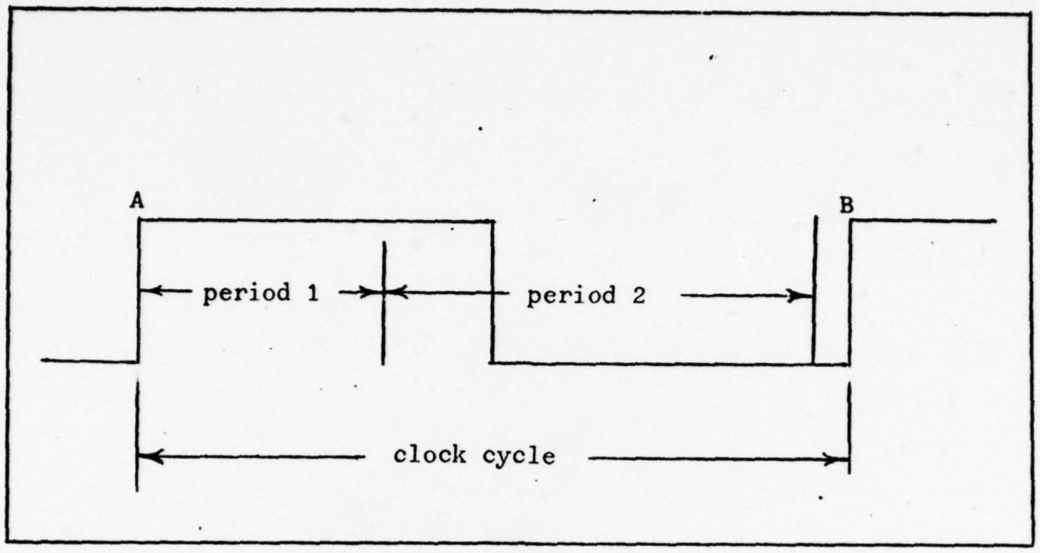


Figure A5. Micro Level Transfers

and is executed during the clock cycle from A to B. Concurrently, the address of the next microword is calculated during *period 1* and applied to the MDB. The addressed microword is fetched from CS during period 2 and clocked into the PL at point B. Thus, one microinstruction is executed and another fetched during each clock cycle.

Macro Level Transfers. Macro level transfers are accomplished using three 16 bit buses: DB, AB, and MB. As shown in Figure A1, the DB is the transfer path between registers in the CCU, ALU, MEM, and I/O modules. The MB and AB are used for transfers to and from memory. Control of macro transfers is accomplished using control signals derived from the PL. As shown in Figure A6, these control signals are available shortly after the beginning of any particular clock cycle until the end of the cycle. This allows ALU functions to be computed and routed to the destination register during a single clock cycle.

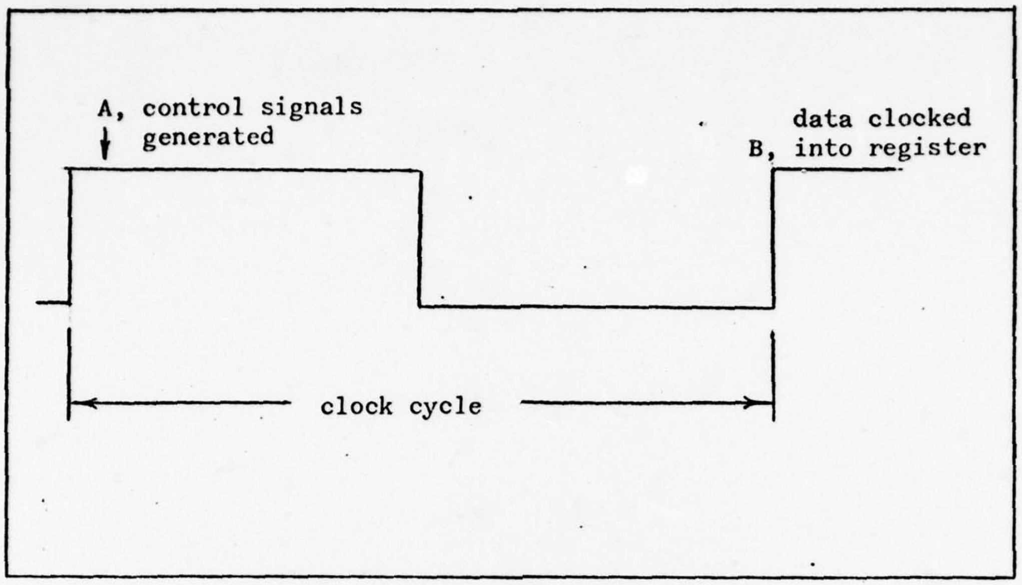


Figure A6. Macro Level Transfers

Module Descriptions

Front Panel Module. Figure A7 depicts a block diagram of the Front Panel designed for MIME, while Figure A8 presents the Front Panel as it appears to the user. Appendix 1 of this manual contains detailed schematics of the two front panel cards, FP1 and FP2.

FP1 contains all of the front panel switches, displays, and logic shown in the block diagram, except the Display Multiplexer (MUX) logic, which is contained in FP2. Consequently FP1 is attached directly to the hinged chassis front panel, while FP2 is mounted in the card rack.

As shown in Figures A7 and A8 the Front Panel logic is grouped in the same manner as the FP1 switches and displays. Separate logic submodules are used for each of the primary front panel functions: Control, Macro Display, Data Entry, and Micro Display. The Display Mux logic is provided to allow selection of any MIME bus for display. Front panel operation will be discussed from the viewpoint of the specific occurrances resulting from each front panel switch activation.

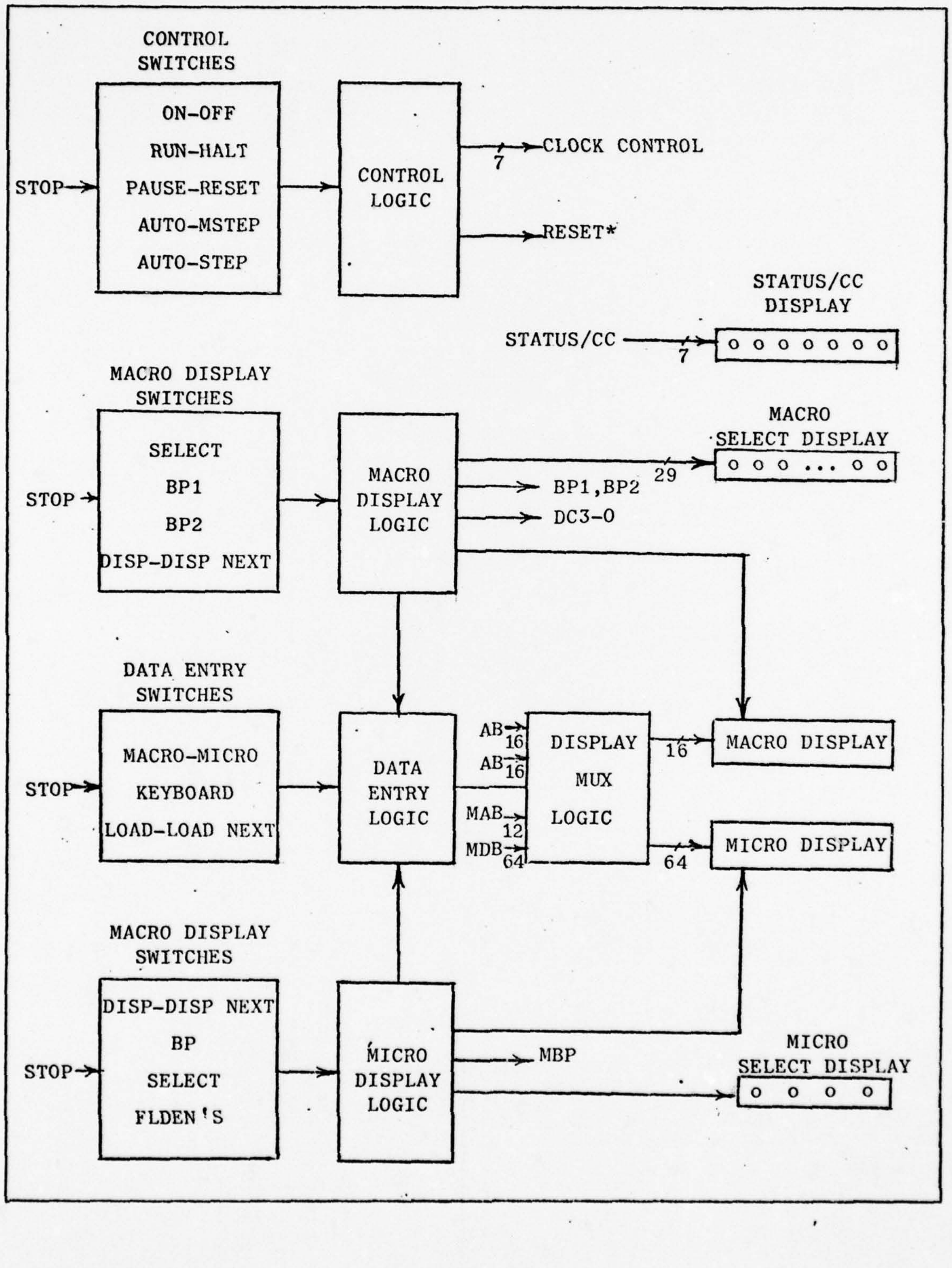


Figure A7. Front Panel Block Diagram

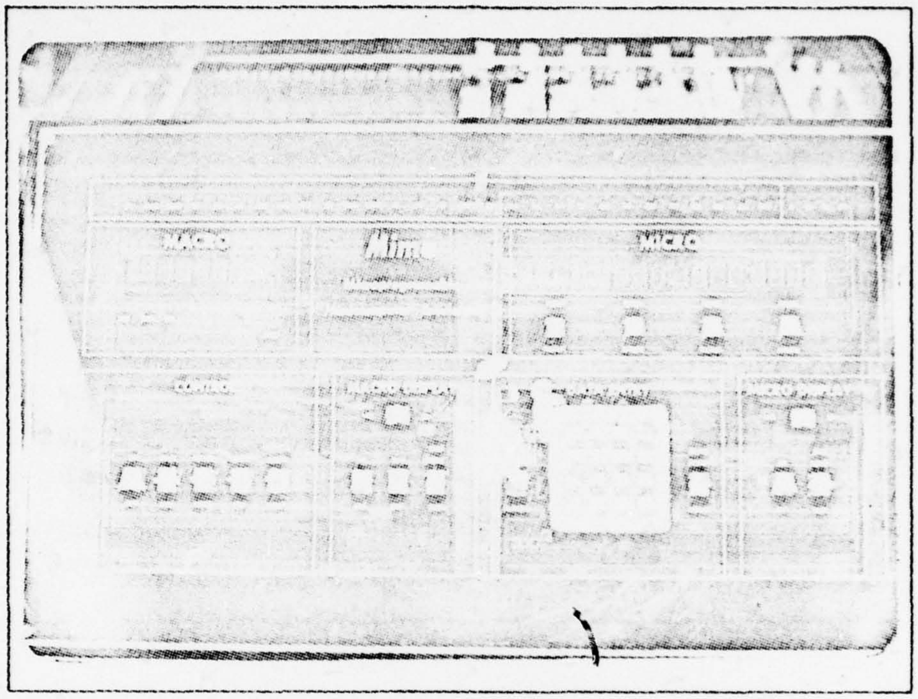


Figure A8. MIME Front Panel

CONTROL. Turning the ON-OFF switch ON applies power to the built in power supplies (when installed). Turning the switch off removes power.

Momentarily depressing RUN causes the CCU to synchronously place MIME in the RUN mode on the next high-to-low master clock transition. Similarly, HALT causes the CCU to synchronously place MIME in the HALT mode when MIME enters the instruction fetch routine.

Momentary PAUSE synchronously halts MIME on the next high-to-low master clock transition, and resets the clock mode to fast. Momentary RESET generates the master clear pulse RESET*. RESET* is also generated during power-up and accomplishes the following:

1. places MIME in HALT mode
2. loads initial microaddress

3. clears PC, MEMFF, MAR, WCR, BAR, CCR, DBR

4. selects PC and A for macro and micro display respectively

If MIME is halted, a momentary STEP causes only a single machine instruction to be fetched and executed before MIME is returned to the HALT state. Similarly, a momentary AUTO STEP causes successive machine instructions to be fetched and executed once each 1.5 seconds. MIME will then halt under program control or when HALT, PAUSE, or RESET is momentarily activated.

Initial actuation at MSTEP changes the clock mode to manual. Subsequent MSTEP's can then be used to single-step through microroutines. AUTO MSTEP provides a 0.7 Hz clock. MSTEP and AUTO MSTEP should not be activated while MIME is running in the fast clock mode.

MACRO DISPLAY. The Macro Display Select switch designated in Figure A8 by < > selects the particular macro level register to be loaded or displayed by the front panel. The selected register is indicated by the illuminated LED above the macro display. Loading/displaying the MBR loads/displays the memory location addressed by the MAR. Loading the ALU loads the ALU DBR, while displaying the ALU displays the ALU output. Since Q, Ri, and BPi are located within the ALU, the ALU DBR must first be loaded to load these registers. Q, Ri and BPi may be displayed directly. For operator convenience, R14 and R15 are designated BP1 and BP2 on the front panel. Identification of the selected register is encoded into a four bit Destination Code (DC) which is routed to the CCU and ALU. A fifth bit is used to differentiate between the sixteen Am2901 general purpose registers and the discrete dedicated registers (e.g. IR, PC, etc.).

BP1 and BP2 enable the macro level breakpoint registers. When enabled, MIME is halted by the CCU whenever the PC matches the contents of BP1 or BP2. This is accomplished under firmware control by comparing the PC with the contents of the enabled breakpoint register(s) and halting MIME if they are equal.

DISP gates the contents of the selected register onto the DB/AB and then into the macro display register. DISP NEXT accomplishes the same function and also increments the MAR if MBR is the selected register. This allows convenient display of sequential memory locations. Both DISP and DISP NEXT are operational only in the HALT mode. The selected register is automatically displayed after each STEP, AUTO STEP, MSTEP, or AUTO MSTEP cycle.

DATA ENTRY. Macro and micro level programs are loaded from the front panel using the hexadecimal (hex) KEYBOARD. Successive hex digits are shifted into the least significant (rightmost) macro or micro display as specified by the MACRO/MICRO switch. Each character of the selected display shifts one position to the left as new data is entered; the leftmost character is last.

Contents of the selected display are loaded into the specified macro/micro level register by momentarily depressing LOAD. LOAD NEXT does the same, as well as incrementing the MAR (macro) or A (micro).

MICRO DISPLAY. The Micro Display switch selects the micro level register to be loaded or displayed by the front panel. The micro-address register (A) and micro breakpoint register (BP) are 12 bits long and require only three hex displays; the remaining displays are blanked out. Control Store (CS) and Pipeline (PL) are 64 bits long, requiring the entire micro display. BP controls the micro breakpoint feature, which halts MIME when the microaddress matches the contents of the micro breakpoint register. DISP displays the contents of the selected micro level register, whereas DISP NEXT also increments the microaddress if CS is displayed. The selected micro level register is automatically displayed after each MSTEP or AUTO MSTEP cycle.

Each of the FLD ENA switches selectively enables one quarter of the micro display, allowing any quarter(s) of the display (and CS) to be independently changed. Enabling any particular field (four hex characters) of the display disables each NORM field. The entire display is enabled by placing all four switches in either the NORM or FLDEN positions.

ALU. Figure A9 depicts the ALU configuration and a schematic of a schematic of the ALU module is included in Appendix 1. As shown in Figure A9, the 16 bit ALU is based on two identical 8 bit byte processors with shifting linkages, condition code selection, and other ancillary functions provided by associated multiplexers, registers, and buffers. This configuration allows autonomous upper and lower byte arithmetic, logic, or shifting operations as well as full word operations when the two byte processors are simultaneously enabled. Each byte processor consists of two Am2901 four-bit bipolar microprocessor slices having combined resources of a 16 word by 8 bit two-port RAM, an 8 bit Q register, an 8 bit ALU element, and associated shifting, decoding, and multiplexing circuitry. The Am2901 is fully described in Reference 3.

The ALU Data Buffer Register (DBR) accepts data from the 16 bit DB on the low-to-high transition of CP1 when ALULD* is low, and it is then immediately available at the D inputs of all Am2901's. A and B addresses for the two-port RAM are available from the AMUX and BMUX1

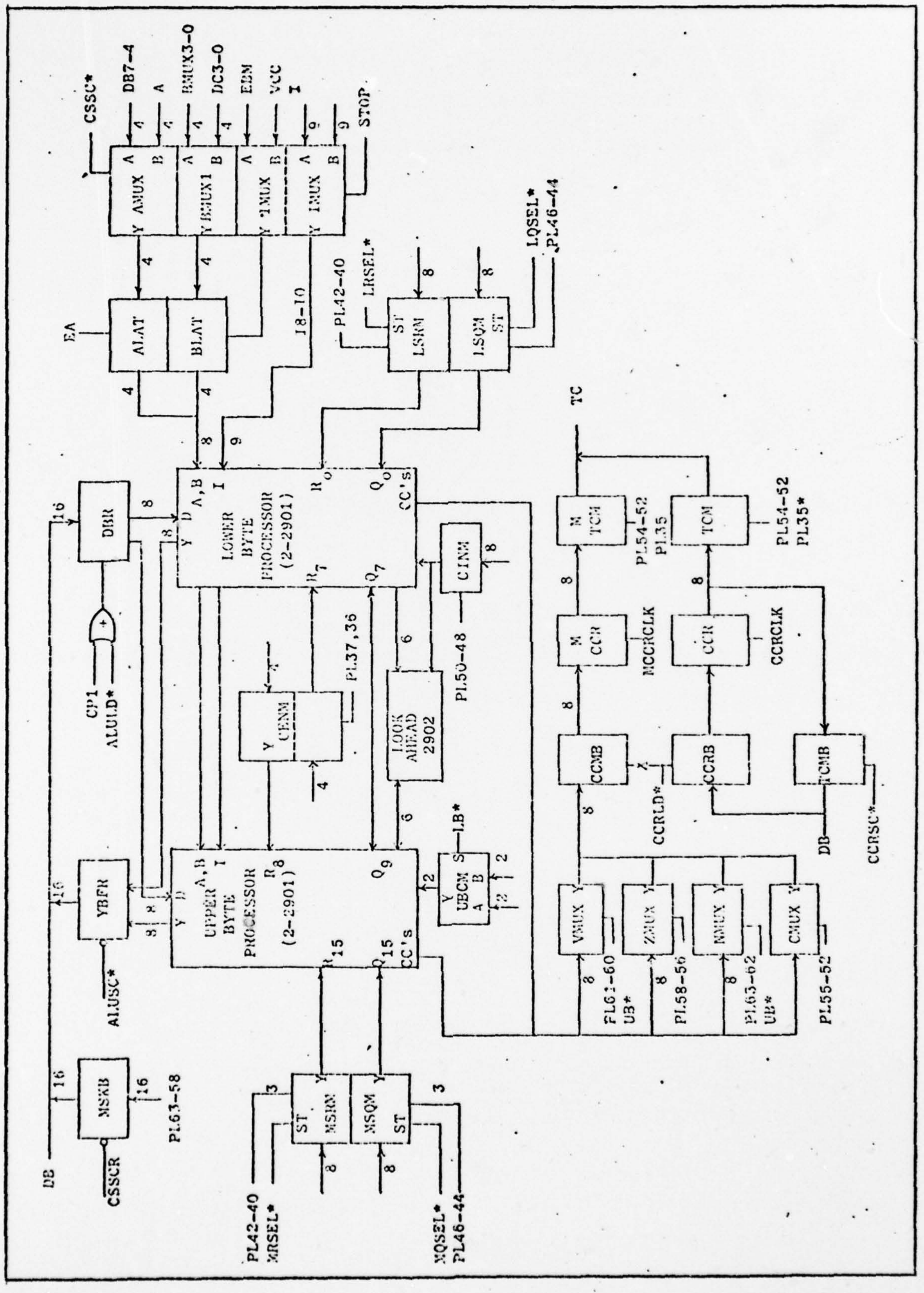


Figure A9. ALU Block Diagram

and are latched by ALAT and BLAT respectively. As specified by CSSC*, the AMUX selects the A address from either the PL or DB, and it is latched into ALAT by EA on the low-to-high transition of CP1 when either ALTLD* or ABLTLD* is low. BMUX1 and BMUX2 (see CCU module description) select the B address from the PL, DB, or front panel Display Code *DC) as specified by CSSC* and STOP. The B address is latched into BLAT during the RUN mode when either BLTLD* or ABLTLD* is low. BLAT is transparent.

Nine microinstruction bits are required to control processor operations. These bits enter the ALU module through the Instruction Multiplexer (IMUX) to provide front panel load and display capabilities while in the STOP mode. Both byte processors receive the same control bits and hence perform the same function. Autonomous byte operations are possible because only the processor whose clock is enabled will store the resultant data in its internal RAM or Q register. Data generated by the nonenabled processor is lost.

Byte processor construction allows a one bit right or left shift of computational results during each clock cycle before storage in the internal RAM or Q registers. Ri and Qi connections serve as tri-state bidirectional ports for the most and least significant bits of each byte. Shift linkages to these ports are provided by an array of tri-state multiplexers. During right shifts, the Most Significant RAM Mux (MSRM) selects the scurce of RAM bit 15 while the Most Significant Q Mux (MSQM) selects the source of Q bit 15. During left shifts, the sources of RAM bit 0 and Q bit are selected by LSRM and LSQM respectively. The Center Mux (CENMUX) is used to select the source of RAM bit 8 during left shifts and RAM bit 7 during right shifts.

Q bits 7 and 8 are directly connected. Microinstruction bits used to control shift linkage multiplexers are discussed in the microprogramming section. Note that MRSEL* and MQSEL* are generated by CCU to prevent inadvertant enabling of MSRM and MSQM during left shifts and likewise for right shifts.

Data is output to the DB from the Am2901 Y outputs via the YBFR when ALUSC* is low, and it is clocked into the destination register by the next CP1 low-to-high transition.

The Carry-In Mux (CINM) selects an appropriate value to be the look-ahead carry inputs for the upper byte during word operations and ripple carry inputs during byte operations. The overflow (V), zero (Z), negative (N), and carry (C) condition codes may be selected from the results of arithmetic operations, or they may be set or cleared using the VMUX, ZMUX, NMUX, or CMUX multiplexers. A less than B (ALB), B less than A (BLA), and the LSB of the ALU output (Y0) are also available as condition codes. Condition codes may be stored in either the macro (CCR) or micro (MCCR) level condition code register when PL35 is low or high respectively.

Multiplexers TCM and MTCM select the particular condition code to be output as a test condition. To facilitate interrupt handling, the CCR is gated onto the DB by TCRB when CCRSC* is low. MSKB is provided for gating PL63-56 on to the DB when CSSC* is low as a means of transferring masks and constants stored in the microword to other MIME modules.

MEMORY. As shown in Figure A10, the MEM1 module contains 1K words of RAM, the MAR, the MBR, and the byte write logic. Information transfers necessary for read and write cycles in the run mode will be explained. Stop mode operation will also be discussed.

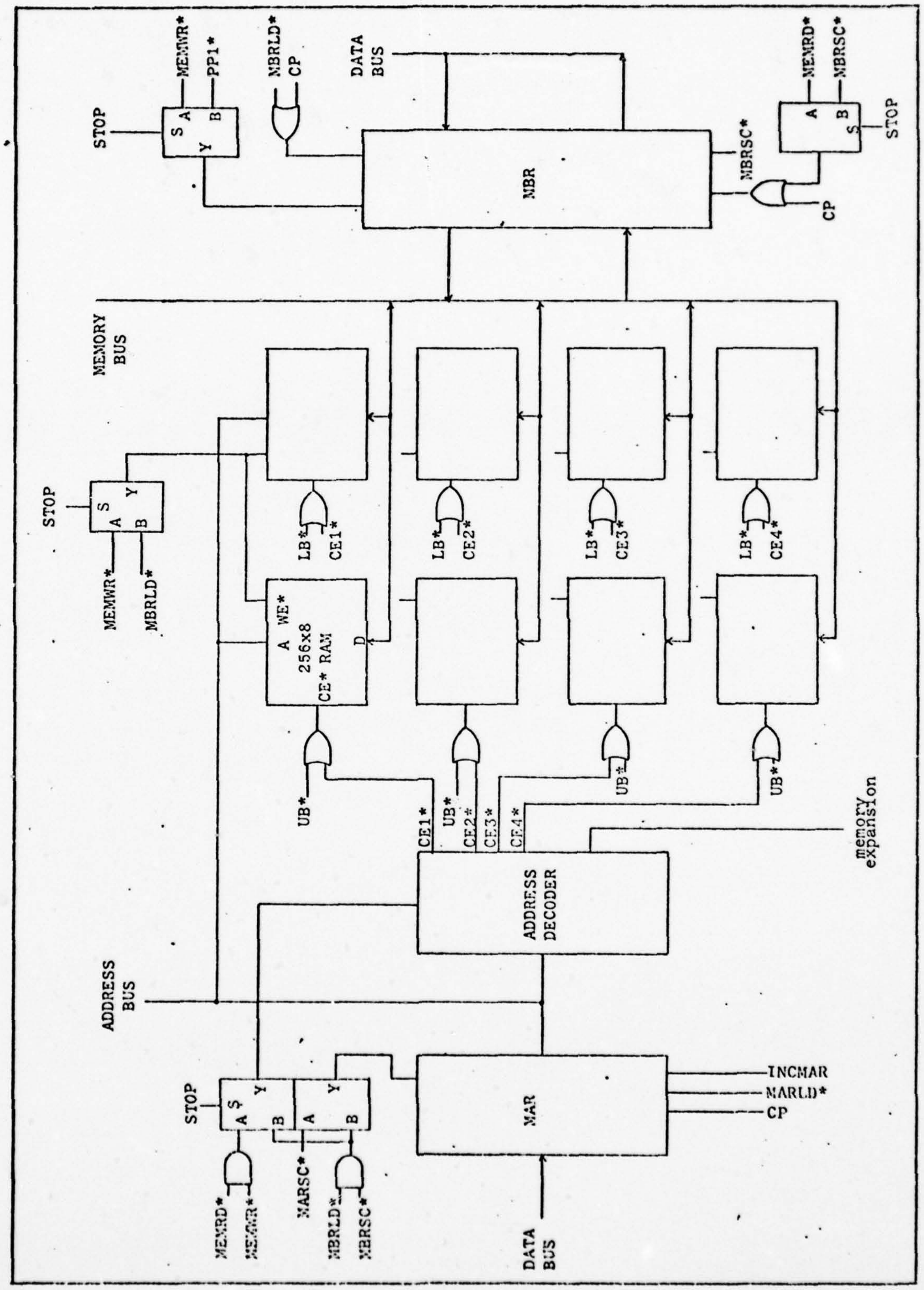


Figure A10. MEM1 Block Diagram

Data which is to be written into memory must first be loaded into the MBR from the DB using signal MBRLD*. Next, the effective memory address must be loaded into the MAR using MARLD*. A memory write cycle is then initiated using MARSC*, UB* and/or LB*, and MEMWR*. MARSC* gates the MAR contents onto the address bus, allowing it to be decoded to enable the appropriate 256x8 RAM block(s) as indexed by the byte enable signals UB* and LB*. MEMWR* gates MBR data onto the MB and into the addressed word of the enabled RAM block(s).

A memory read cycle is preceeded by loading the MAR as in a write cycle. The read cycle is then initiated using MARSC*, UB* and/or LB*, and MEMRD*. Address decoding is accomplished as in a write cycle, and MEMRD* is used to load the contents of the addressed work into MBR from the MB.

Memory data is loaded from the front panel in the stop mode by selecting MBR and depressing LOAD or LOAD NEXT. This gates an address onto the AB, clocks data into the MBR, and gates MBR onto the MB and into the addressed location. LOAD NEXT increments the MAR when released.

A memory location is similarly displayed from the front panel. Depressing DISP or DISP NEXT gates an address onto AB, clocks memory data from MB into MBR and gates MBR onto the DB for FP display. Both UB* and LB* are held low while stopped allowing only full word front memory manipulations.

<u>COMPUTER CONTROL UNIT</u>. The CCU block diagram of Figure A11 depicts the CCU which is based upon Am2911/AM2909 microsequencers (Ref 2) and has associated logic used for initiating address generation and test condition selection. Also included are the IR, PC, PL, and Micro Breakpoint Register as well as control signal decoders and master

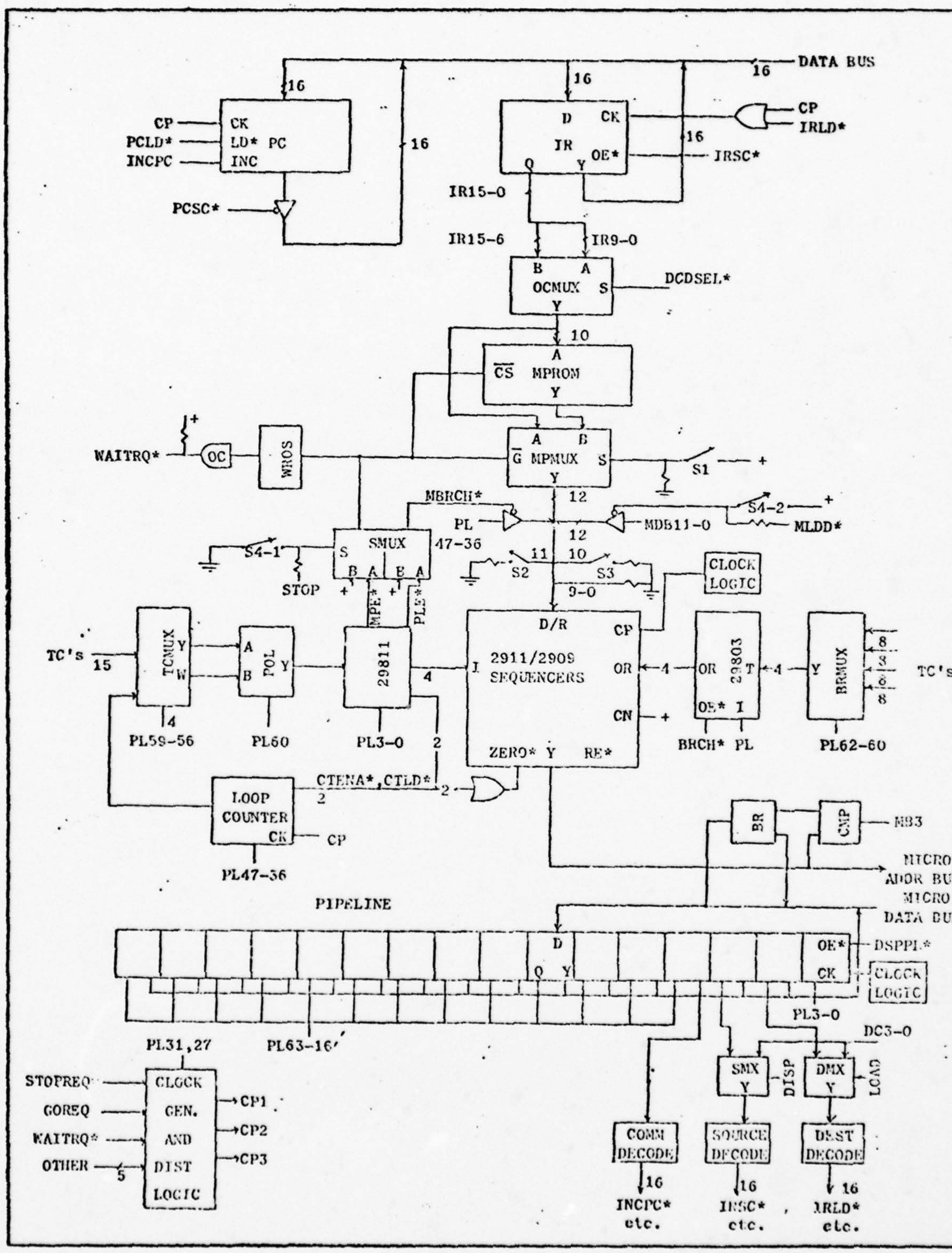


Figure A11. CCU Block Diagram

clock generation/distribution logic. CCU operation will be explained by examining typical transfers occurring chronologically during the fetch and decode of a single machine instruction.

Initially, the PC will be either incremented by INCPC or loaded from the DB by PCLD*. This provides the address of the next machine instruction. Control signal PCSC* will then be used to gate this address onto the DB for transfer to the memory. A memory read cycle yields the next instruction which is loaded from the DB into the IR on the next low-to-high clock transition when IRLD* is low. As controlled by DCDSEL*, OCMUX selects the op code portion of the IR and applies it to the address lines of the Mapping PROM (MPROM). MPROM contains a unique microroutine starting address for each particular op code thus allowing flexible CS organization. S1 controls MPMUX which allows the user to bypass MPROM and use the op code directly as the microroutine starting address.

The MPMUX tri-state outputs are one of three starting address sources. The other two sources are PL bits 47-36 and DB bits 11-0. The PL bits are used during micro-branching operations and the DB bits are used to load a starting address from the front panel.

Regardless of the source, the starting address is loaded into the Am2911/Am2909 microsequencer D/R inputs, routed to the Y output, and then gated onto the MAB. The corresponding microword is fetched from CS and loaded into the PL on the next low-to-high clock transition. These PL bits are used as control signals throughout MIME. Three of the four-bit PL fields are decoded to provide 16 control signals each. To provide front panel load and display capabilities, the BUS SRC and BUS DST fields are routed through multiplexers SMX and DMS respectively and then decoded.

Addresses of subsequent microwords to be fetched from CS are generated by the microsequencers as controlled by PL bits 3-0. The Am2911/Am2909 architecture allows micro level straight line programming as well as conditional branching, looping, and subroutining. Which of these that occurs is controlled by 1 of 16 test conditions selected by the TCMUX and POLMUX. The selected test condition and PL bits 3-0 are inputs to the AM29811 which in turn controls the microsequencers, MPROM starting address source, and Micro Loop Counter (MLC).

When loaded with a value N, the MLC allows N+1 executions of a micro level loop. Reference 2 contains a detailed discussion of the Am29811 as well as the Am29803 discussed below (Ref 2:1-9 to 1-15, 2-11 to 2-21).

The Am29803 provides a method of simultaneously testing up to four condition codes selected by BRMUX and then branching to the microaddress specified by the test result. This is accomplished using the branch table technique discussed in section III of this manual.

Also included in the CCU is clock generation and distribution logic. This logic generates the master clock CP1 and two ALU clocks, CP2 and CP3. The latter two are controlled by PL or front panel control signals that specify the desired ALU operation.

Several features have been provided specifically for checkout and troubleshooting purposes. As previously discussed, S1 allows the user to bypass the MPROM during checkout of new microroutines or during module tests. Switch S4 inhibits front panel loading of starting addresses in order to allow the complete Am29811/29803 capabilities to be used for troubleshooting. Switches S2 and S3 allow the lowest address of the selected 1K block of CS to be used as the starting address when MIME is initialized or manually reset.

CONTROL STORE. Figure A12 depicts the CS1 module which contains 1K microwords of EPROM and 256 microwords of RAM. As in the memory module, CS read and write cycles will be discussed.

In the run mode, one CS read operation is accomplished each clock cycle. The address generated by the CCU is decoded to enable one block of RAM or EPROM. The addressed word of the enabled block is routed through the Control Store Buffer (CSB) onto the MDB to be clocked into the PL.

CS is not presently writable in the run mode. Control line CSW may be used to write to CS under program control providing appropriate hardware buffers are added between data source and MDB.

Data may be written to the CS RAM in the stop mode by selecting CS and using LOAD or LOAD NEXT. Signal MDBOFF is low with CS selected. Depressing LOAD (NEXT) causes MDBOFF to go high, allowing CS portions selected by front panel FLDEN switches (Wi*low) to be written into. MDBOFF inhibits nonselected CS portions (Wi*high) by disabling corresponding CSB portions.

CS data is displayed from the front panel by depressing DISP or DISP NEXT. This latches the CS output into the micro display. Selecting PL, A, or MBP on the front panel causes MDBOFF to go high, disabling the CSB and isolating CS1 from the MDB.

I/O. The I/O module of Figure A13 contains the Am2914 Vectored Priority Interrupt Controller, the BAR, the WCR, I/O TC MUX, and Direction flip-flop (DFF).

The Am2914 (Ref 4) is used to control eight levels of vectored priority interrupt I/O operations. Interrupt requests are accepted on eight lines, and signal IRQ* is pulled low if the highest priority non-masked request is greater than or equal to STR contents. The sixteen

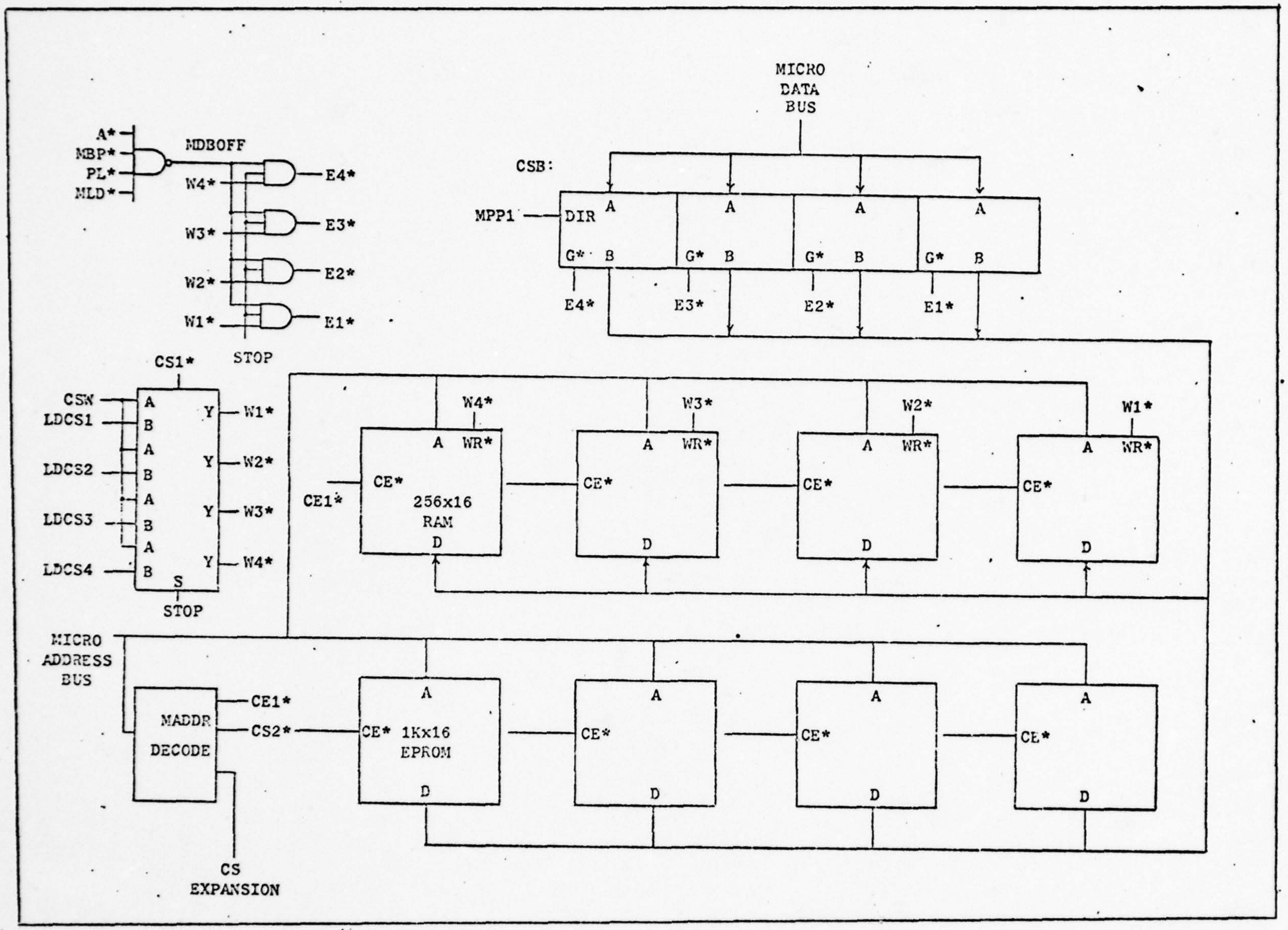


Figure A12. CS Block Diagram

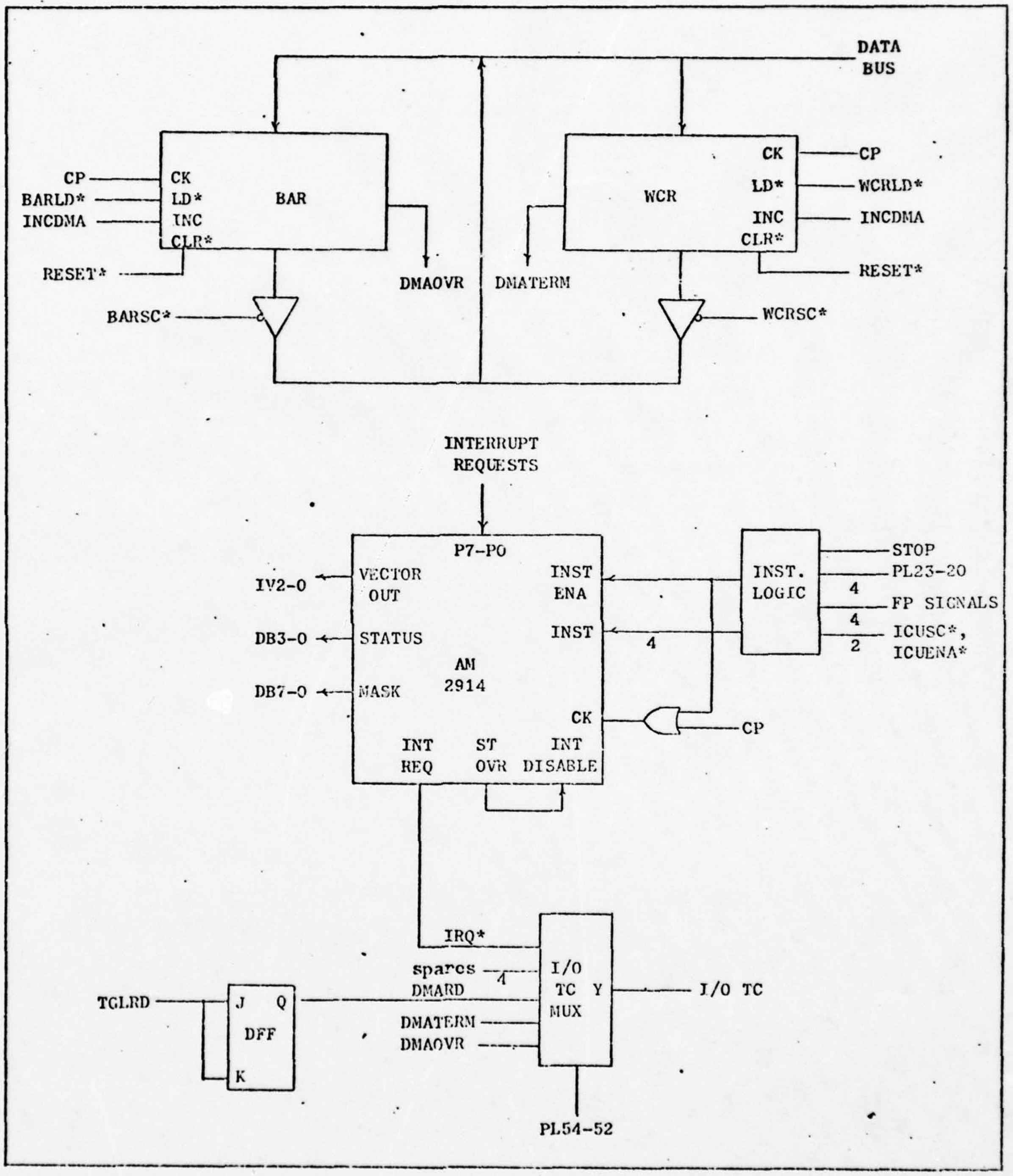


Figure A13. I/O Block Diagram

AM2914 instructions are summarized in the microprogramming section of this manual. Both the SR and MR may be loaded under program control with these instructions, or from the front panel. The instruction logic interfaces the AM2914 to the front panel and prevents inadvertent contention on the DB.

The BAR is loaded with the initial memory address while the WCR is loaded with the two's complement of the number of words to be transferred. The BAR is gated onto the AB for each DMA cycle, and then both the BAR and WCR are incremented. The DMATERM flag is used to indicate completion of the desire number of transfers. The DMAOVR flag is set if the memory address overflows.

The I/O TC MUX selects 1 of 8 I/O related condition codes for testing by the CCU. One of these codes is DMARD, the DFF output. DFF is loaded with the transfer direction during DMA initialization and is tested to determine direction of each subsequent DMA transfer.

CHASSIS DETAILS. The chassis was designed to provide maximum access to MIME components. The front panel is hinged (restricted to 130° travel) to allow access to FP1 wiring. Removal of the plexiglass faceplate allows access to all FP1 components. The remaining circuit cards are rack mounted and access to components and wiring may be obtained by card removal. An extender card has been provided to allow access to one card under operating conditions. Access to the backplane has been provided by a removable panel on the bottom of the MIME cabinet.

For user convenience, a Miscellaneous Signal Panel (MSP) has been provided which contains a HEX/OCTAL select switch (macro display), power connections and test points. The HEX/OCTAL switch determines whether data displayed on the macro level is in hexadecimal or octal format. The power connections wer provided as a means of powering MIME with laboratory power supplies until internal supplies are installed. The power connections may then be used as a power source for external experimental setups. The test points which are discussed under trouble-shooting were provided to allow easy access to dynamic signals of interest.

II. Operation

This section presents procedures used to load and execute macro and micro level programs. It is assumed that the user is familiar with the System Description section of this manual. Capitalized words refer to FP1 switches and indicators.

Macroprogram Loading

Machine language programs are loaded into MEM from FP1 in the following manner:

1. Select Hex/Octal format (on MSP)
2. Select MAR (macro level register select)
3. LOAD MAR with initial memory address (select macro level and enter value via KYBRD and LOAD)
4. Select MBR (macro level register select)
5. LOAD (NEXT) memory location with data

Microprogram Loading

CS contains both read-only and read/write memory for microinstruction storage. Read-only CS can be changed by removing and reprogramming the CS EPROM. Read/write CS can be loaded from FP1 using the following procedure:

1. Select A (micro level register select)
2. Load initial microaddress (select micro level and enter value via KYBRD and LOAD)
3. Select CS (micro level register select)
4. LOAD (NEXT) CS location with data

Program Execution

The following procedure is used to execute MIME programs:

1. Place CCU switches 1, 2, and 3 in desired positions (select CS block, see CCU Module Description)

2. RESET MIME to initialize microaddress

3. LOAD PC with address of first machine instruction (macro level register select enter value via KYBRD and LOAD)

4. Select macro level register for auto display (STEP, AUTO STEP, MSTEP, or AUTO MSTEP modes)

5. Select micro level register for auto display (MSTEP and AUTO STEP modes)

6. Begin program execution in desired mode by use of one of the following:

    a. RUN

    b. STEP

    c. AUTO STEP

    d. MSTEP, RUN, MSTEP

    e. AUTO MSTEP, RUN

7. Program execution may be halted under program control or by activating one of the following:

    a. HALT

    b. PAUSE

    c. RESET

III. Microprogramming

This section presents information which enables the reader to implement machine instructions as microinstruction modules. The microword fields (related groups of bits) are defined first, and because many of these fields control multiple functions (i.e. they are formatted or serve more than one function), representative microword formats are then presented. As an example, these concepts are then applied to the emulation of several PDP-11/03 instructions.

MICROWORD FIELDS

A total of 150 signals are used to control MIME. A 150 bit microword would have been extremely flexible but prohibitively expensive when realized in hardware. Therefore, the techniques of encoding and formatting were used to reduce the microword length to a more reasonable 64 bits. As shown in Figure A14, fields 2, 3, and 4 are encoded four bit fields which each provide 16 control signals through use of 1 of 16 decoders. The majority of the other fields are formatted, such that they each control two or more functions as specified by a set of steering bits. Each field is discussed in detail below, and summarized in Tables AIII to AX.

MICROADDRESS SEQUENCER. This field controls microaddress sequencing using the Am29811 next address control unit in conjunction with Am2909/11 microinstruction sequencers (Ref 2:1.9-2.21). One of the sixteen Am29811 instructions shown in Table AIII must be specified in each microinstruction.

BUS DESTINATION. This field is decoded to control gating of DB information into 1 of 10 registers. As shown in Table AIV, it also controls MEM read and write cycles (in conjunction with UB* and LB* fields and MARS bus source).

CONSTANT												
BR MUX		BR CONTROL		I/O TC MUX			MICRO LOOP COUNTER LOAD					
	POL	CCU TC MUX		ALU TC MUX			MICRO BRANCH ADDRESS					
N MUX	V MUX		Z MUX	CARRY MUX		CINMUX		Q MUX		R MUX		CEN MUX
63-62	61-60	59	58 - 56	55 - 52	51	50 - 48	47	46 - 44	43	42 - 40	39 38	37-36

						I/O					
CC*	ALUDST	UB*	ALUFN	LB*	ALUSRC	ALU A ADDR	ALU B ADDR	COMMAND	BUSSRC	BUSDST	SEQ MADDR
35	34 - 32	31	30 - 28	27	26 - 24	23 - 20	19 - 16	15 - 12	11 - 8	7 - 4	3 - 0

Figure A14. Microword Definition

Table AIII

Microaddress Sequencer Field

MICROADDRESS SEQUENCER							
MNE-MONIC	MICRO CODE	INSTRUCTION	TEST INPUT	NEXT ADDR SRC.	FILE	CNTR	ENA (1)
JZ	0	Jump to address zero	X	D	HOLD	-	PL
CJS	1	Conditional jump-to-subroutine; PL address	L H	MPC D	HOLD PUSH	HOLD HOLD	PL PL
JMAP	2	Jump to Mapping PROM Output	X	D	HOLD	HOLD	PROM
CJP	3	Conditional jump to PL	L H	MPC D	HOLD HOLD	HOLD HOLD	PL PL
PUSH	4	Push file; conditionally load counter	L H	MPC MPC	PUSH PUSH	HOLD LOAD	PL PL
JSRP	5	Jump-to-subroutine; address conditionally R or PL	L H	R D	PUSH PUSH	HOLD HOLD	PL PL
CJV	6	Conditional jump to initial micro address (CCU S2 and S3)	L H	MPC D	HOLD HOLD	HOLD HOLD	N.A. N.A.
JRP	7	Jump to address; conditionally R or PL	L H	R D	HOLD HOLD	HOLD HOLD	PL PL
RFCT	8	Repeat loop if counter ≠ 0. (Must test CT*)	L H	F MPC	HOLD POP	DEC HOLD	PL PL
RPCT	9	Repeat PL if counter ≠ 0. (Must test CT*)	L H	D MPC	HOLD HOLD	DEC HOLD	PL PL
CRTN	A	Conditional return-from-subroutine	L H	MPC F	HOLD POP	HOLD HOLD	PL PL
CJPP	B	Conditional jump to PL and POP File	L H	MPC D	HOLD POP	HOLD HOLD	PL PL
LDCT	C	Load counter and continue	X	MPC	HOLD	LOAD	PL
LOOP	D	Repeat loop if condition ≠ 1	L H	F MPC	HOLD POP	HOLD HOLD	PL PL
CONT	E	Continue to next addr.	X	MPC	HOLD	HOLD	PL
JP	F	Jump to PL address	X	D	HOLD	HOLD	PL

(1) Notes: ENA gates specified data to D/R 2911/09 inputs. D may be selected as next address source on same clock cycle. R is loaded on clock ↑ and may be selected on next clock cycle. MPC contains current address +1. F = file output. PL = microbranch field.

Table AIV

Command, Bus Source, and Bus Destination Fields

COMMAND			BUS SOURCE			BUS DESTINATION		
MNEM	CODE	FUNCTION	MNEM	CODE	FUNCTION	MNEM	CODE	FUNCTION
IPC	Ø	Increment PC.	PCS	Ø	Gate PC onto DB.	PCL	Ø	Load PC from DB.
IMAR	1	Increment MAR.	MARS	1	Gate MAR onto AB.	MARL	1	Load MAR from DB.
IDMA	2	Increment WCR & BAR	MBRS	2	Gate MBR onto DB.	MBRL	2	Load MBR from DB.
TGRD	3	Toggle READ FF.	IRS	3	Gate IR onto DB.	IRL	3	Load IR from DB.
BRCH	4	Enable 16-way branch	MSTP	4.	Halt Command	MEMR	4	Write to MEM from MBR
RSTM	5	Reset Memory FF.	-	5	Spare	MEMW	5	Read from MEM to MBR
ALD	6	Load A address latch	ALUS	6	Gate ALU output onto DB	DBRL	6	Load DBR from DB.
BLD	7	Load B address latch	CCRS	7	Gate CCR onto DB.	CCRL	7	Load CCR from DB.
ABLD	8	Load A and B latches	-	8	FP display of MKR	-	8	FP load of MKR.
DCDS	9	Decode lower part of IR.	ICUS	9	Enable I/O IMC, RSR, RMR.	-	9	FP load of STR
-	A	Spare	BARS	A	Gate BAR onto DB.	BARL	A	Load BAR from DB.
-	B	Spare	WCRS	B	Gate WCR onto DB.	WCRL	B	Load WCR from DB.
ICU	C	Enable Am 2914	FPS	C	Gate FP data onto DB.	FPL	C	Load FP disp. from DB.
IOOT	D	Write to I/O device*	CSS	D	Gate CONSTANT onto DB.	-	D	Spare
IOIN	E	Read from I/O	IOBS	E	Gate IOBR onto DB*	IOBL	E	Load IOBR from DB*
NOOP	F	No Operation	NOOP	F	No Operation	NOOP	F	No Operation

* See Ref 6

BUS SOURCE. The bus source field, also shown in Table AIV, gates data onto the DB from 1 of 9 registers, and onto the AB from 1 of 2 registers. Both the bus source and destination fields must be specified for each microinstruction. In addition, memory read/write cycles require bus source MARS or BARS to gate addresses onto the AB. Bus source IOBS must be selected during specific Am2914 instructions discussed in the interrupt control field section.

COMMAND. The command field generates 1 of 16 miscellaneous control signals as summarized in Table AIV. This field must be specified for each microinstruction.

ALU A and B Address. These two field serve as a source of the ALU two-port RAM A and B addresses. A and B addresses must be loaded into the ALU ALAT and BLAT respectively using command mnemonic ALD, BLD, or ABLD. These latches, as discussed in the ALU module description, may also be loaded from the DB.

INPUT/OUTPUT. This field uses the instructions shown in Table AV to control the Am2914 Vectored Priority Interrupt Controller (Ref 4). This field must be used in conjunction with command ICU. In addition, bus source ICUS must also be used with IMC, RSR, and RMR to avoid multiple DB sources.

ALU SOURCE, FUNCTION, DESTINATION. As shown in Table AVI, these three fields control the ALU Am2901's. ALU source selects 1 of 8 operand pairs; function selects 1 of 8 operations; and destination controls shifting and destination of processing results.

UB* and LB*. These two 1 bit fields control upper and lower byte ALU and MEM storage respectively. As shown in Table AVI, ALU storage is enabled by selecting the appropriate ALU destination field with UB* and/or LB* low. MEM write is accomplished with MEMW, MARS, and UB*

Table AV

Input/Output Field

INTERRUPT CONTROL UNIT		
MNEM	CODE	INSTRUCTION
MCL	0	Master clear: clear all interrupts, MR , and SR ; Enable interrupt requests
CAI	1	Clear all interrupts
DBC	2	Clear interrupts from DB data
IMC	3	Clear interrupts from MR (uses DB)
CLV	4	Clear interrupt associated with last vector read
RDV	5	Read vector (IV2-0); load SR +1 into SR
RSR	6	Gate SR onto DB
RMR	7	Gate MR onto DB.
SMR	8	Set MR (inhibits all interrupts)
LSR	9	Load SR from DB
BCM	A	Bit clear MR from DB
BSM	B	Bit set MR from DB
CMR	C	Clear MR (enables all interrupts)
DIR	D	Disable interrupt request
LMR	E	Load MR from DB
EIR	F	Enable interrupt request

Table AVI

Am2901 Processor Fields

ALU DST			
MNEM	CODE	Y	RAM/Q STORAGE
QOF	0	F	F → Q
OF	1	F	no storage
BOA	2	(A)	F → B
BOF	3	F	F → B
RBQ	4	F	F/2→B,Q/2→Q
RB	5	F	F/2 → B
LBQ	6	F	2F→B, 2Q→Q
LB	7	F	2F → B

ALU FN		
MNEM	CODE	ALU FUNCTION
ADD	0	$R + S + C_{in}$
S-R	1	$S + R^* + C_{in}$
R-S	2	$R + S^* + C_{in}$
OR	3	$R \vee S$
AND	4	$R \wedge S$
NRAS	5	$R^* \wedge S$
XOR	6	$R \veebar S$
XNOR	7	$(R \veebar S)^*$

ALU SRC			
MNEM	CODE	SOURCE OPERANDS	
		R	S
AQ	0	A reg	Q reg
AB	1	A reg	B reg
ØQ	2	Ø	Q reg
ØB	3	Ø	B reg
ØA	4	Ø	A reg
DA	5	DBR	A reg
DQ	6	DBR	Q reg
DØ	7	DBR	Ø

CCU*		
MNEM	CODE	FUNCTION
CC	0	Load CCR
MCC	1	Load MCCR

UB*		
MNEM	CODE	FUNCTION
ON	0	Enable upper byte store, ALU and MEM
OFF	1	Disable upper byte store, ALU and MEM

LB*		
MNEM	CODE	FUNCTION
ON	0	Enable lower byte store ALU and MEM
OFF	1	Disable lower byte store ALU and MEM

and/or LB* low. ALU storage may be disabled during MEM write with ALU destination OF. UB* and LB* must be specified for each microinstruction.

CC*. CC* determines whether the condition codes generated as the result of arithmetic operations are latched by the CCR (CC* low) and the MCCR (CC* high) as shown in Table AVI.

MICRO LOOP COUNTER LOAD. As shown in Figure A14, the Micro Loop Counter Load field is a formatted field. The contents of this field is loaded into the MLC when the Microaddress Sequencer field contains PUSH or LDCT. M-1 is loaded to accomplish N loops.

MICROBRANCH ADDRESS. As shown in Figure A14, the Microbranch Address field is also formatted. The contents of this field conditionally specifies the address of the next microinstruction when the Microaddress Sequencer field contains CJB, CJP, JSRP, JRP, RPCT, CJPP, or JP.

CENMUX, RMUX, QMUX. These fields are used with the ALU destination field to select appropriate shifting linkages as shown in Table AVII.

I/O TC, ALU TC, CCU TC, POL. As shown in Table AVIII, these fields select condition codes for testing by the CCU. I/O TC selects 1 of 8 condition codes generated in the I/O module, while ALU TC selects 1 of 16 generated in the ALU. Eight of the ALU condition codes are stored in the CCR and eight in the MCCR. CCU TC selects 1 of 16 condition codes, including those preselected by the I/O TC and ALU TC. POL specifies positive or negative test logic.

CARRY, ZERO, OVR, NEG. These fields select sources for ALU condition codes CARRY, ZERO, OVR, and NEG. As shown in Table AIX, UB* is used with OVR and NEG to automatically select the correct byte when NRM is specified.

Table AVII

Shifting Linkage Fields

QMUX			
		LEFT SHIFT	RIGHT SHIFT
MNEM	CODE	QØ	Q15
RRO	0	RAM 15	RAM Ø
QRO	1	Q 15	QØ
C	2	CARRY	CARRY
CLR	3	GND	GND
SET	4	VCC	VCC
–	5	SPARE	SPARE
D15	6	SPARE	DBR 15
R8	7	SPARE	RAM 8

RMUX			
		LEFT SHIFT	RIGHT SHIFT
MNEM	CODE	RAMØ	RAM15
RRO	0	RAM 15	RAM Ø
QRO	1	Q 15	QØ
C	2	CARRY	CARRY
CLR	3	GND	GND
SET	4	VCC	VCC
Q7	5	Q7	SPARE
D15	6	SPARE	DBR 15
ROB	7	RAM 7	RAM 8

CENMUX			
		LEFT SHIFT	RIGHT SHIFT
MNEM	CODE	RAM 8	RAM 7
CLR	0	GND	GND
C	1	CARRY	CARRY
D7	2	SPARE	DBR 7
NRM	3	RAM 7	RAM 8

Table AVIII

Test Condition Fields

POL		
MNEM	CODE	Polarity
POS	0	Positive
NEG	1	Negative

CCU TC MUX		
MNEM	CODE	CC
BP1	0	BP1
BP2	1	BP2
I/O	2	I/O TC
ALU	3	ALU TC
MARØ	4	MARØ
MEM	5	MEM
CT*	6	CT*
IR4	7	IR4
IR5	8	IR5
IR7	9	IR7
IR15	A	IR15
–	B	spare
–	C	spare
–	D	spare
–	E	spare
set	F	VCC

ALU TC MUX		
MNEM	CODE	ALUTC
–	0	Spare
MFØ	1	MFØ
MAL	2	MALBT
MAG	3	MAGBT
MC	4	MCRY
MV	5	MOVR
MZ	6	MZERO
MN	7	MNEG
–	8	spare
FØ	9	FØ
AL	A	ALBT
AG	B	AGBT
C	C	CARRY
V	D	OVER
Z	E	ZERO
N	F	NEG

I/O TC MUX		
MNEM	CODE	I/O TC
DMAO	0	DMAOVR
DMAT	1	DMATERM
DMAR	2	DMARD
ST	3	STATUS
–	4	spare
–	5	spare
–	6	spare
IRQ*	7	IRQ*

Table AIX

ALU Condition Code Fields, Carry In Field

NEG MUX		
MNEM	CODE	NEG SOURCE
REP	0	NEG
NRM	1	UN/LN
CLR	2	GND
SET	3	VCC

OVR MUX		
MNEM	CODE	OVR SOURCE
REP	0	OVER
NRM	1	UOVR/LOVR
CLR	2	GND
SET	3	VCC

ZERO MUX		
MNEM	CODE	Z SOURCE
–	0	spare
–	1	spare
LB	2	LZ
UB	3	UZ
REP	4	ZERO
WRD	5	WZ
CLR	6	GND
SET	7	VCC

CARRY MUX		
MNEM	CODE	C SOURCE
C	0	CARRY
C*	1	CARRY*
–	2	spare
–	3	spare
–	4	spare
–	5	spare
AC1	6	AC1
LB	7	LC
UB	8	UC
RØ	9	RAMØ
R15	A	RAM15
Q15	B	Q15
R7	C	RAM7
R8	D	RAM8
CLR	E	GND
SET	F	VCC

CIN MUX		
MNEM	CODE	Carry In
–	0	spare
R7	1	RAM 7
MC	2	MCRY
MC*	3	MCRY*
C	4	CARRY
C*	5	CARRY*
CLR	6	GND
SET	7	VCC

CIN. CIN specifies the ALU carry in during word, upper byte, and lower byte operations. CIN is shown in Table AIX and must be specified for word or byte arithmetic operations.

Branch Control and Branch MUX. These two fields shown in Table AX are used to control the Am29802 16-Way Branch Control Unit (Ref 2:1.13-2.15). The branch control field specifies simultaneous testing of up to four condition codes selected from one of eight sets of four by the branch MUX. The Am29803 output is OR'ed with the four least significant next microaddress bits, resulting in a branch to 1 of 16 sequential locations in CS. Each of the 16 locations of this "branch table" could contain an additional branch instruction, resulting in 16-way branching capability.

Constant. This 16 bit field is gated onto the DB when the bus source field contains CSS, where it is available for use as a constant, ALU RAM A/B address, or interrupt MASK.

Microword Formatting

Twenty four different microword formats are possible using various combinations of the functionally grouped fields. The authors have found that the four general purpose formats shown in Figure A15 are sufficient for most applications.

Arithmetic. The Arithmetic Format makes full use of the data processing capabilities of MIME. Complete arithmetic, logic, shifting, and condition code generation capabilities are available with this format. However, only unconditional microaddress sequencer commands may be used (JZ, JMAP, CONT, and JP).

Table AX

16 Way Branch Fields

BRANCH MUX		
MNEM	CODE	TEST CONDS.
IR0	0	IR3,2,1,0
IR3	1	IR6,5,4,3
IR6	2	IR9,8,7,6
IR9	3	IR12,11,10,9
IR12	4	IR15,14,13,12
IV	5	IV2,1,0
CC	6	Z,N,C,V
–	7	SPARE

BRANCH CONTROL		
MNEM	CODE	TEST
OFF	0	No test
1	1	T0
2	2	T1
3	3	T1,T0
4	4	T2
5	5	T2,T0
6	6	T2,T1
7	7	T2,T1,T0
8	8	T3
9	9	T3,T0
A	A	T3,T1
B	B	T3,T1,T0
C	C	T3,T2
D	D	T3,T2,T0
E	E	T3,T2,T1
F	F	T3,T2,T1,T0

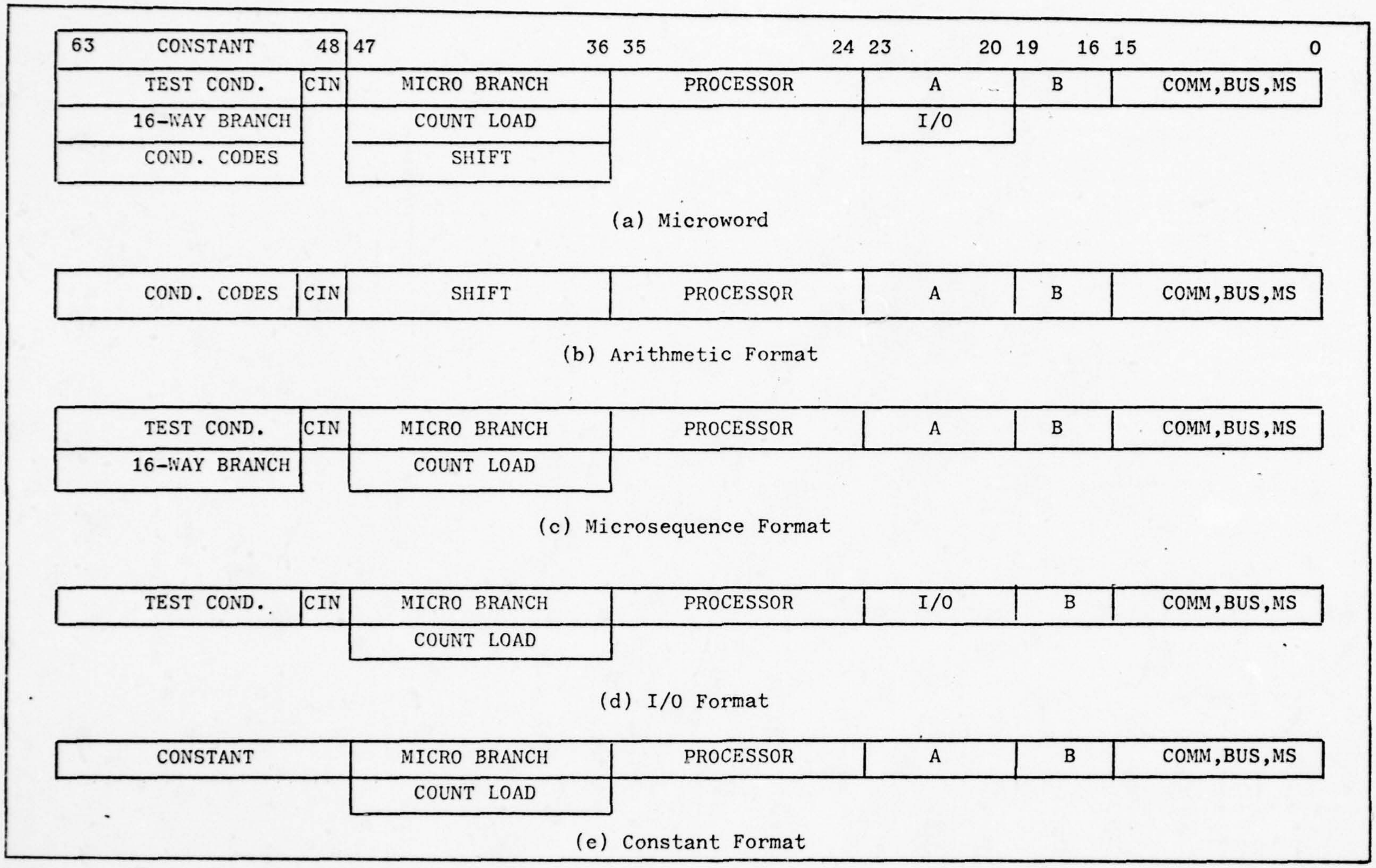


Figure A15. Representative Microword Formats

Microsequencer. This format provides powerful microaddress sequencing features through use of the entire microaddress sequencer instruction set combined with 16-way branching capabilities. The test condition fields are used with the conditional sequencer instructions for micro level branching, looping, and subroutining. In addition, the Am29811 test input is forced high when 16-way branching is enabled, allowing instructions such as jump to 1 of 16 subroutines to be executed.

I/O. The I/O format is used during interrupt processing. The complete sequencer instruction set is utilizable with this format.

Constant. This field is used for transfer of the 16 bit Constant to be DBR, MKR, STR, or A and B address latches. The Am29811 test input is forced high at the same time, providing unconditional versions of the sequencer instructions.

Sample Microprograms

In order to demonstrate the flexibility of microprogramming and its application to emulation, this section presents the emulation of the PDP-11/03 instructions shown in Table AXI using the MIME architecture. Emulation is defined to be the capability of MIME to execute the instructions of another machine with the same processing results.

The authors used the approach discussed in Chapter V of the MIME thesis (Ref 1) to emulate the selected instructions. The discussion of this example consists of two parts: High Level Microprogram Flow and Individual Instruction Modules.

Emulation Structure. The flowchart of Figure A16 shows the microprogram structure used to emulate the selected instructions. Each named block represents a microcode module. Modules above the division line

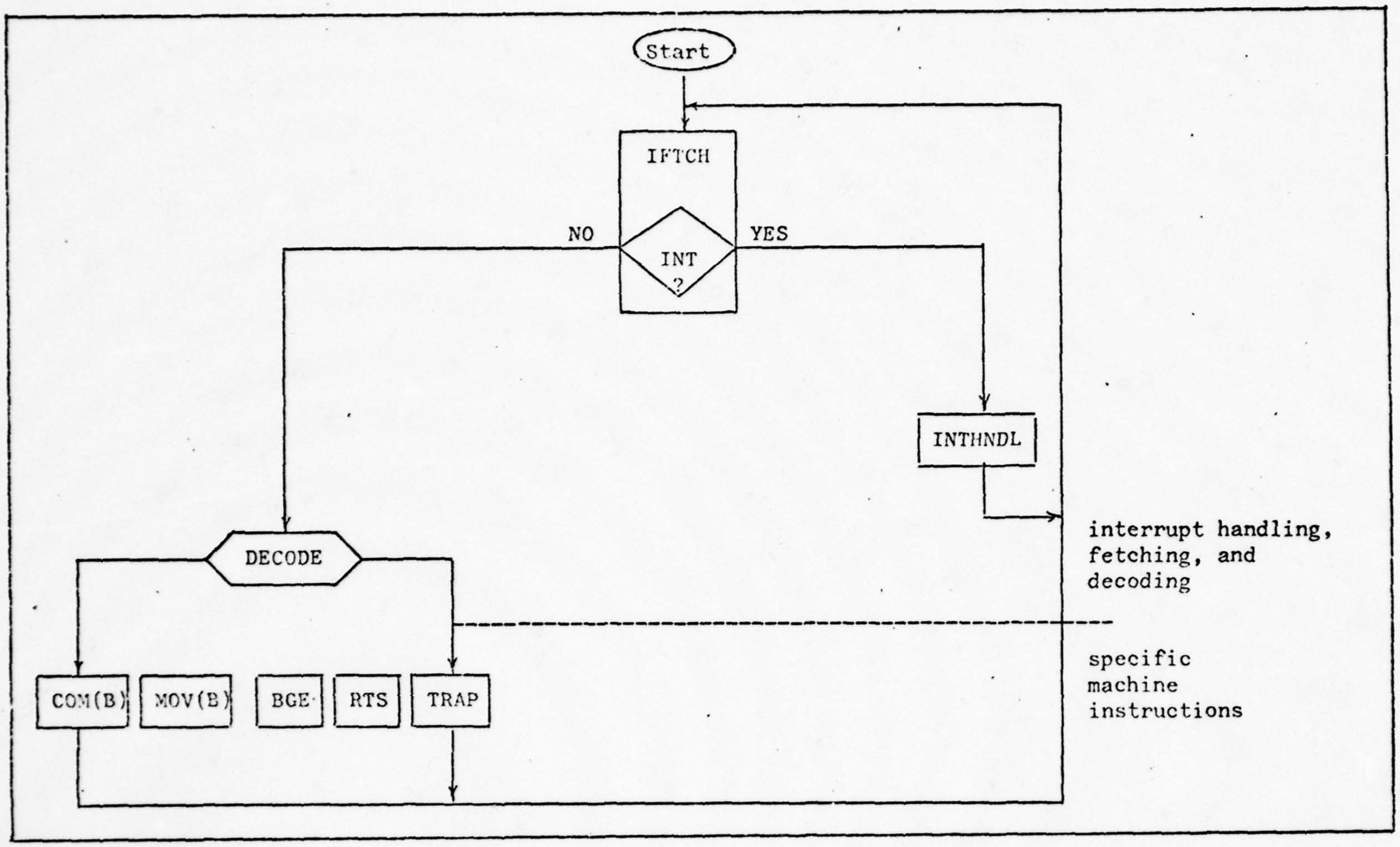

Start
IFTCH
INT ?
NO
YES
INTHNDL
DECODE
COM(B)
MOV(B)
BGE
RTS
TRAP
interrupt handling, fetching, and decoding
specific machine instructions

Figure A16. Emulation Structure

Table AXI

Sample PDP-11/03 Instructions

Instruction Type	Mnemonic	Instruction	Op Code
Single Operand	COM(B)	Complement Destination	/051DD
Double Operand	MOV(B)	Move source to Destination	/1SSDD
Signed Conditional Branch	BGE	Branch if . ≥ 0	002000
Program Control	RTS	Return from subroutine	00020R
Trap	TRAP	Trap	104400 - 104777 / → 1/0

control interrupt handling, fetching, and decoding for all machine instructions. Modules below the line are unique to a particular machine instruction.

After initialization, emulation of a particular machine instruction begins with the Instruction Fetch module (IFTCH) which fetches the next instruction from memory if no interrupts are pending. As shown in Figure A17, IFTCH shifts the PC right one place before loading it into the MAR. This is necessary in order to map the PDP-11/03 memory address space into the MIME address space. The corresponding memory word is then fetched from memory to the MBR and subsequently loaded into the IR. The PC is then incremented by two so that it points to the next sequential instruction.

The DECODE module illustrated in Figure A18 controls the decoding of the instruction op code to arrive at the corresponding micromodule

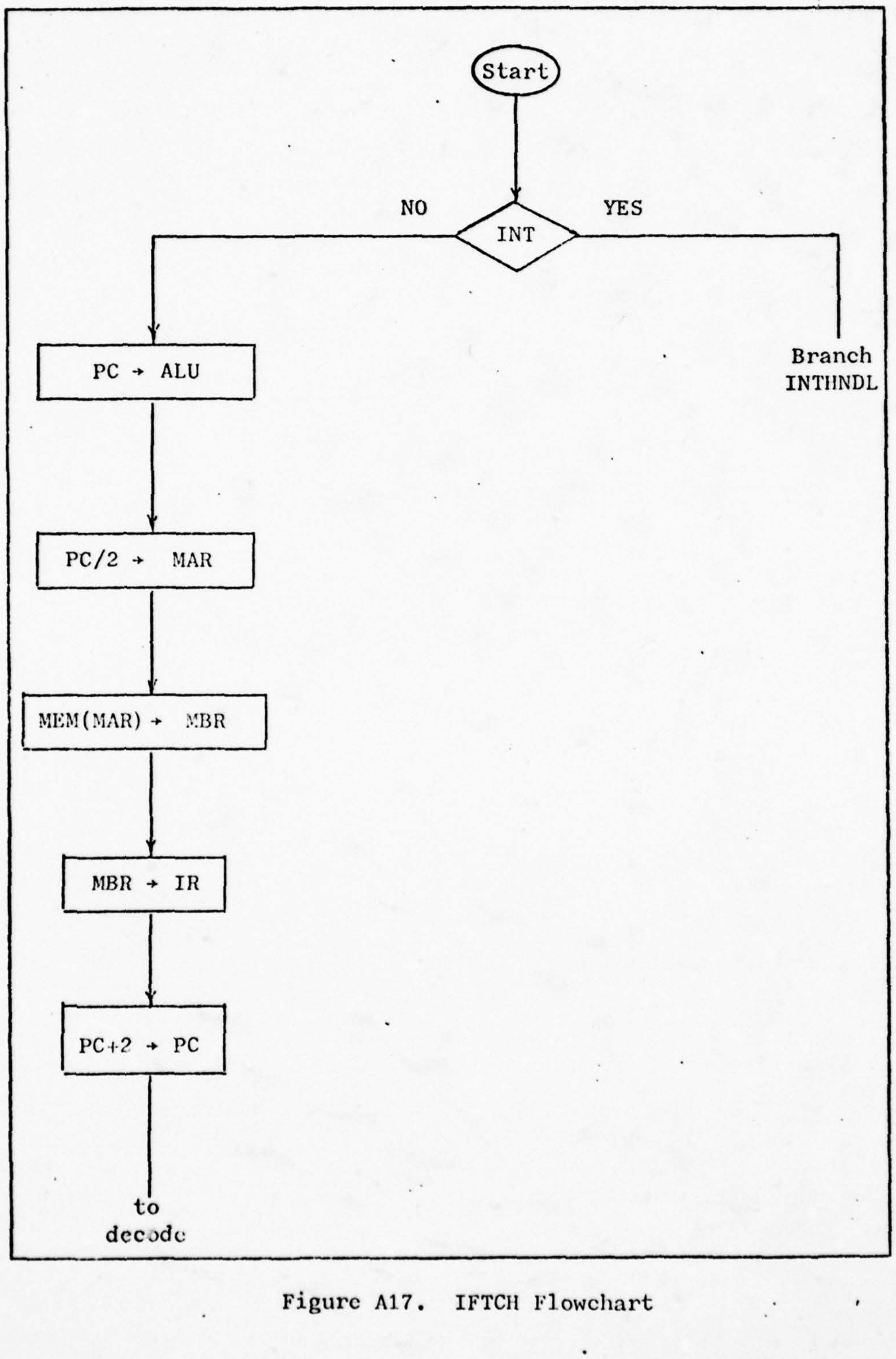


Figure A17. IFTCH Flowchart

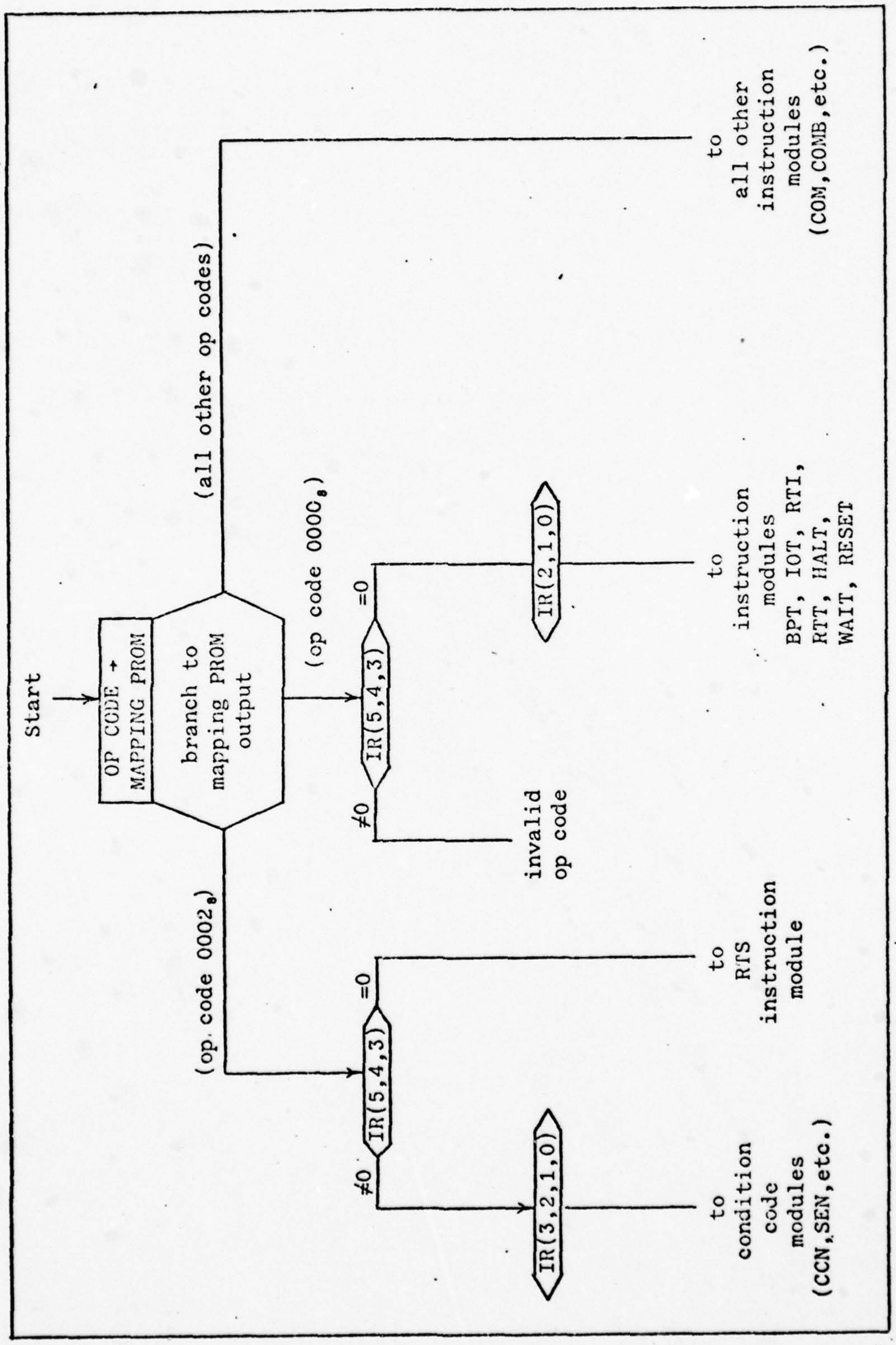

Start
OP CODE → MAPPING PROM
branch to mapping PROM output
(all other op codes)
to all other instruction modules (COM,COMB,etc.)
(op code 0000₈)
IR(5,4,3)
=0
≠0
IR(2,1,0)
to instruction modules BPT, IOT, RTI, RTT, HALT, WAIT, RESET
invalid op code
(op code 0002₈)
IR(5,4,3)
=0
≠0
to RTS instruction module
IR(3,2,1,0)
to condition code modules (CCN,SEN,etc.)

Figure A18. DECODE Flowchart

starting address. This is accomplished in part by using a PROM to map the op code portion of the IR into starting addresses of other micro-modules. If the op code is octal 0002, IR bits 5, 4, and 3 are tested. If IR5,4,3 equals octal 0, a branch to the RTS instruction module is accomplished. Otherwise a branch is made to one of 16 condition code set/clear modules by testing IR bits 3, 2, 1, and 0. If the op code is octal 0000, IR bits 5, 4, and 3 are again tested. In this instance, if IR5,4,3 equals 0, IR bits 2, 1, and 0 are tested to control a branch to one of 8 instruction modules. An invalid instruction has been specified if IR 5,4,3 equals 0. If the op code is other than 0000 or 0002, the mapping PROM output is the starting address of a particular instruction module.

The Interrupt Handler (INTHNDL) module stacks the current PC and CCR in memory and loads a new PC and CCR from pre-assigned memory locations. The PC then contains the address of the first instruction of the machine language interrupt service routine. Referring to Figure A19, this is accomplished by first calling the micro subroutine PUSH. As shown if Figure A20, PUSH decrements R6 (used as the stack pointer) by two, loads the new contents of R6 into the MAR, and returns. The PC is then stored in the memory location addressed by R6. The CCR is similarly pushed onto the stack. The interrupting device is identified by vector IV2,1,0. This vector is used to branch to the particular module necessary to load the MAR with the address of the new PC. The new CCR is in the next sequential location. The PC now points to the machine language interrupt service routine. To execute this instruction, a branch back to IFTCH of Figure A17 is accomplished. After

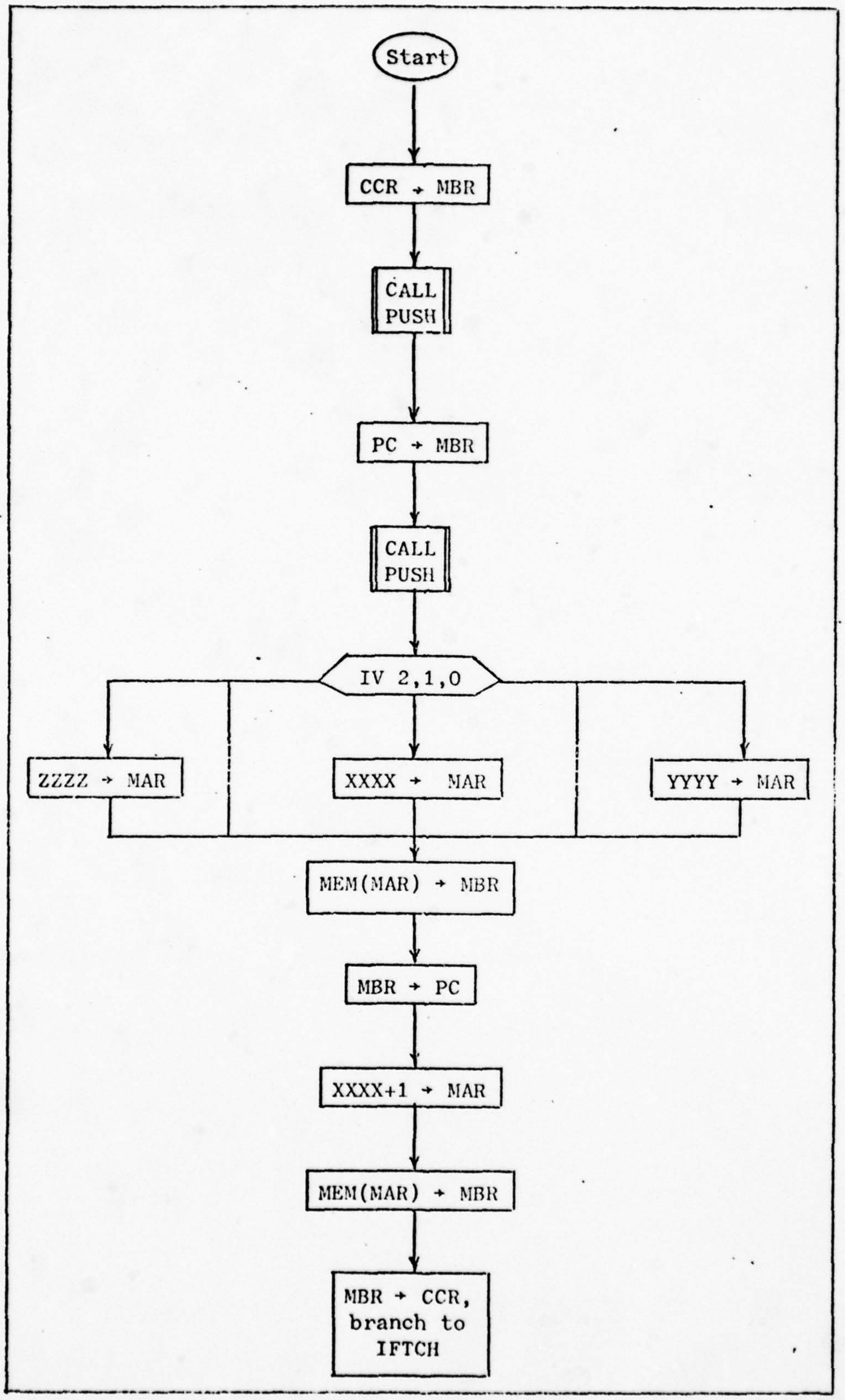


Figure A19. INTHNDL Flowchart

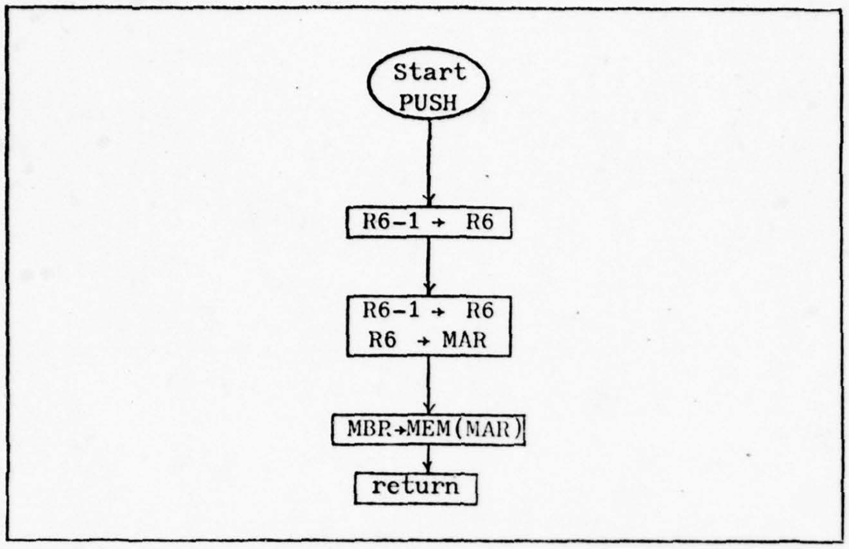


Figure A20. PUSH Flowchart

testing for higher priority interrupts, this instruction is executed using the emulation structure discussed in the preceeding paragraphs.

Individual Instruction Modules. The following paragraphs discuss the micro level modules used to implement the register transfers required for each emulated PDP-11/03 instruction of this example.

COM(B) replaces the word (byte) contents of the destination address with its logical complement. Using the decoding algorithm discussed in the previous section, COM and COMB are decoded to separate starting addresses. Thus, they will be discussed as separate modules. As shown in Figure A21, COM begins by calling the DSTDCD micro subroutine of Figure A22 to compute and load the MAR with the effective address of the operand (memory reference mode). Figure A23 depicts the DPCDCD module used in conjunction with DSTDCD. The reader is referred to the PDP-11/03 Processor Handbook for a comprehensive discussion of instruction formats and addressing modes (Ref 5:3.1-3.19). If the operand is located in memory, micro subroutine OPFTCH shown in Figure A24 is called to fetch the operand and load it into the ALU Data

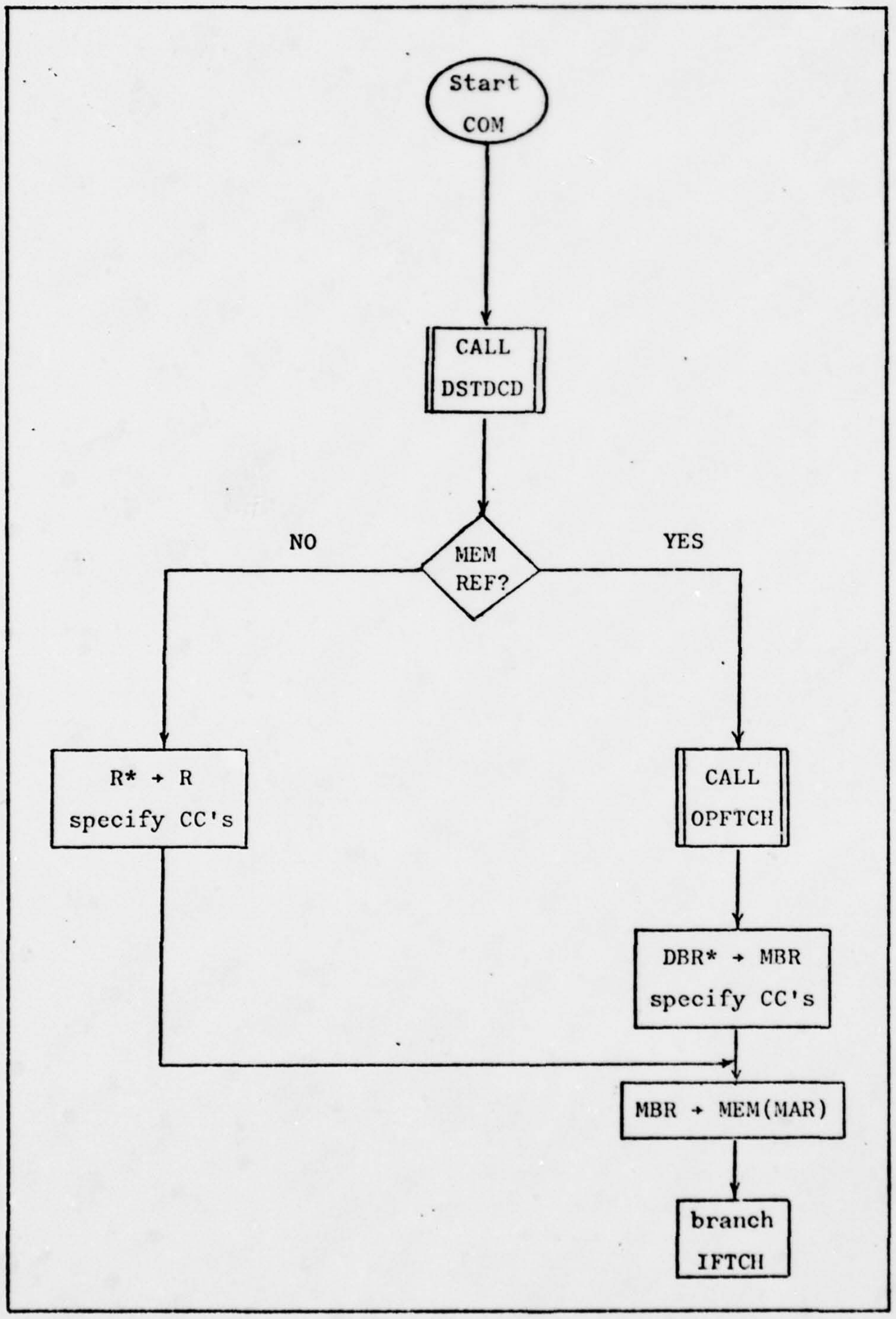


Figure A21. COM Flowchart

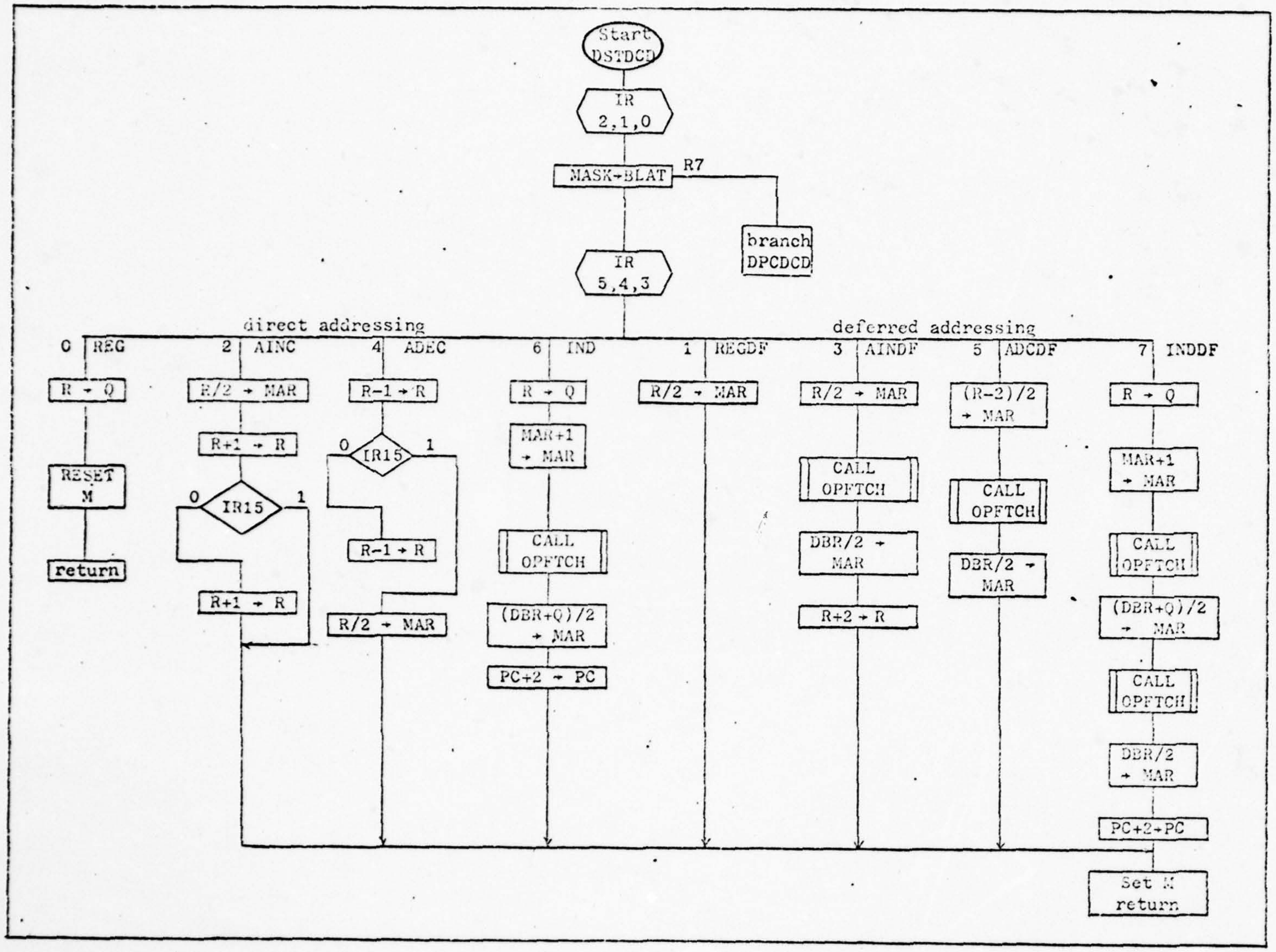


Figure A22. DSTDCD Flowchart

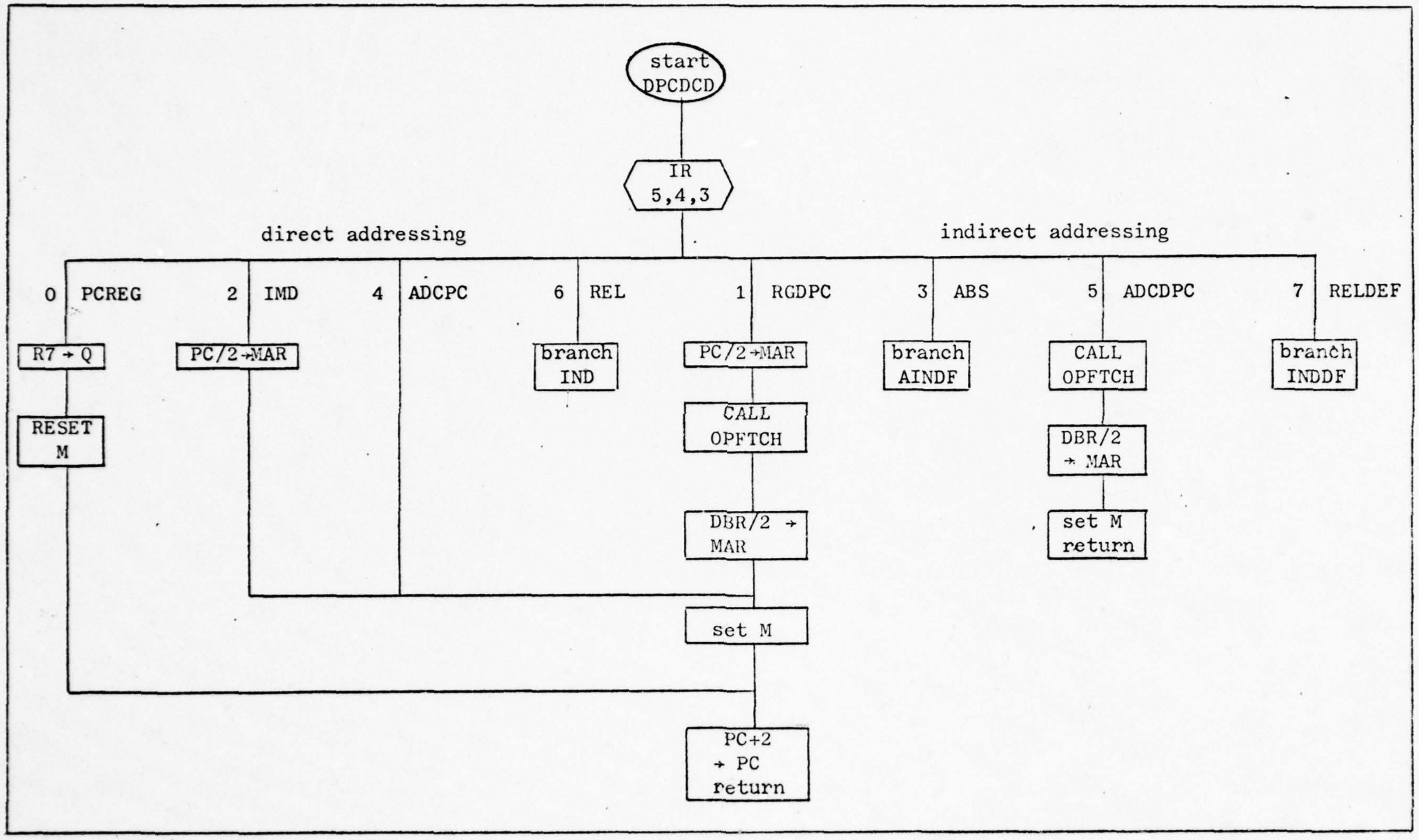


Figure A23. DPCDCD Flowchart

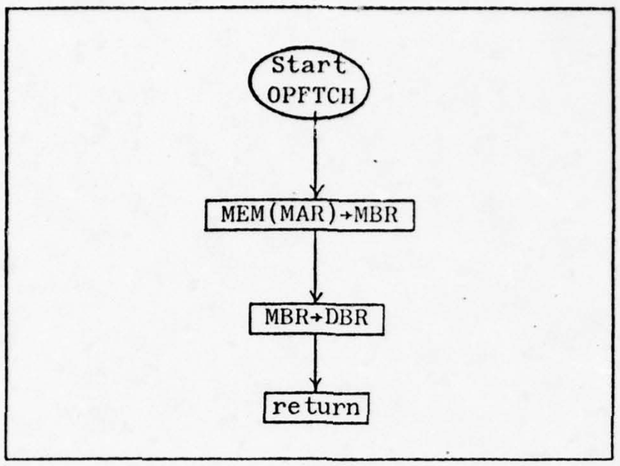


Figure A24. OPFTCH Flowchart

Buffer Register (DBR). The operand is then complemented and loaded into the MBR. Condition codes specified by COM are generated and stored in the CCR at this time. The MBR is then stored in the intial memory location. This completes execution of memory reference COM so the COM module branches back to IFTCH. Register reference COM as shown in Figure A21 is similar to the memory reference, but it omits the memory read and write cycles. CCMB is shown in Figure A25. It differs from COM in that only the specified byte is complemented. Register reference implies lower byte, whereas the least significant MAR bit must be tested to find the applicable memory reference byte.

MOV(B) transfers the source operand word (byte) to the destination location. As shown in Figure A26, MOV calls SRCDCD shown in Figure A27 to find the source operand. OPFTCH is called if memory reference, resulting in the operand being loaded into the ALU DBR. Register reference operands are also loaded into the DBR. The operand is then placed in the Q register and appropriate condition codes generated. DSTDCD is called to determine the destination location, and the operand is stored accordingly. Finally, MOV branches back to IFTCH. As shown

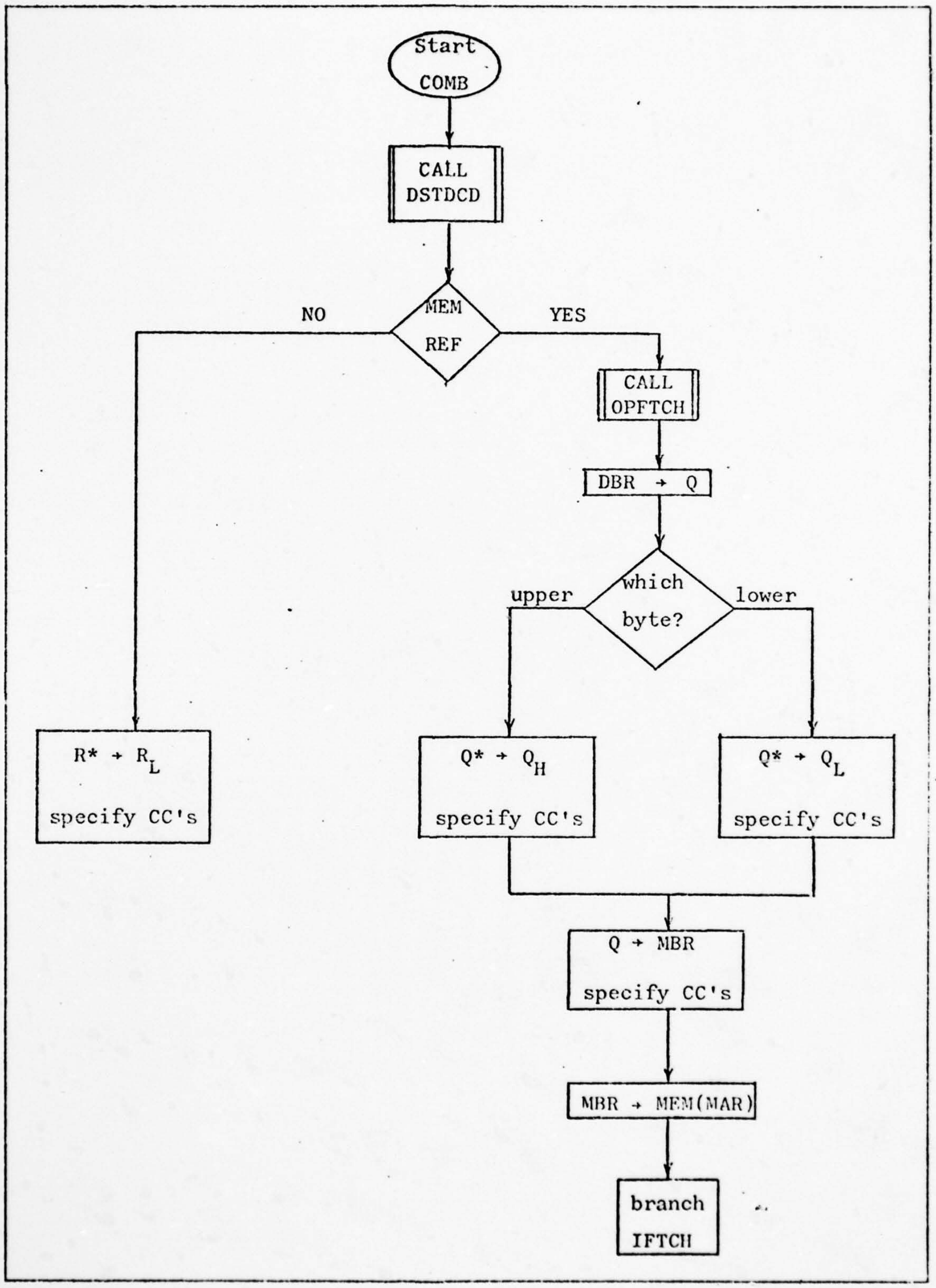


Figure A25. COMB Flowchart

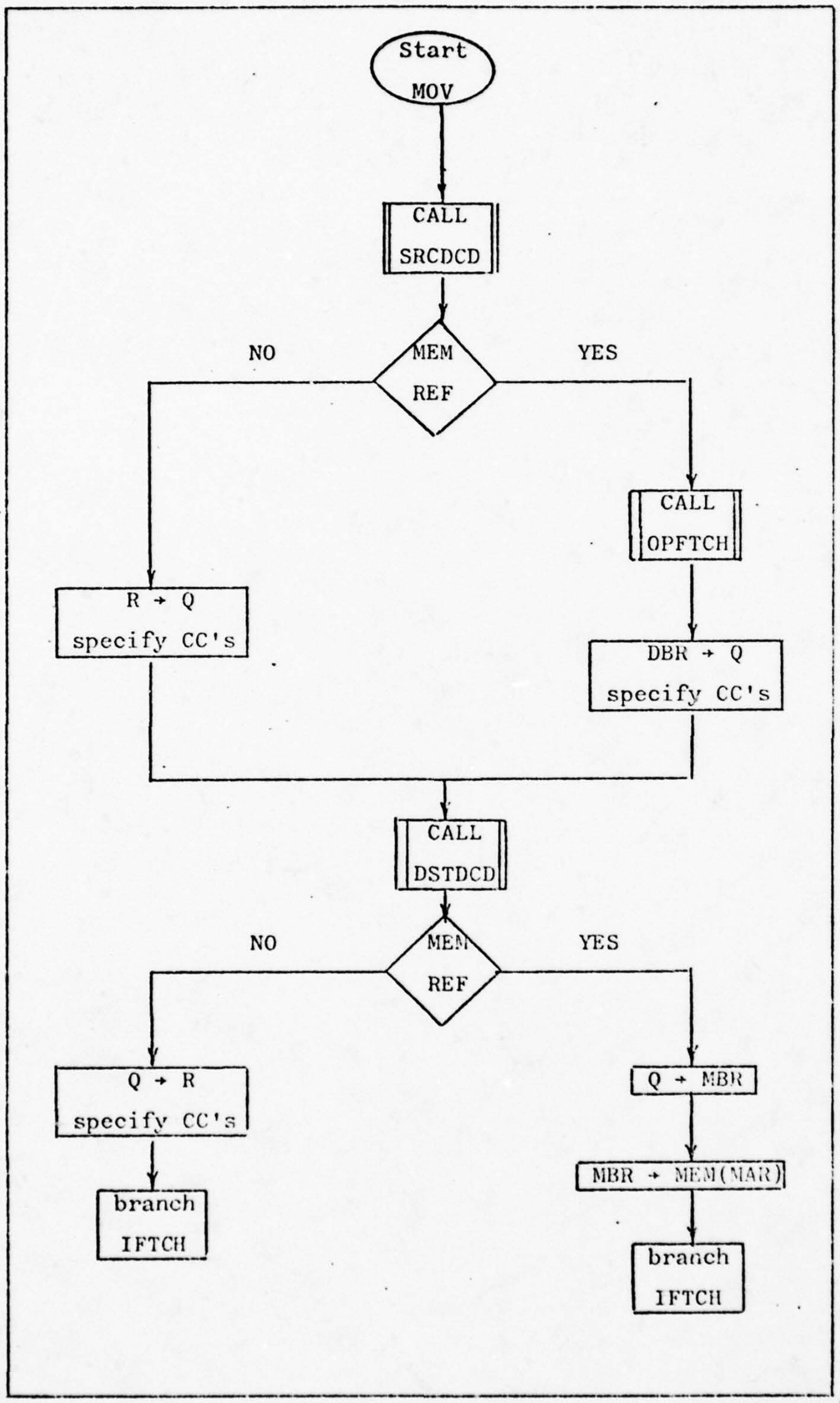


Figure A26. MOV Flowchart

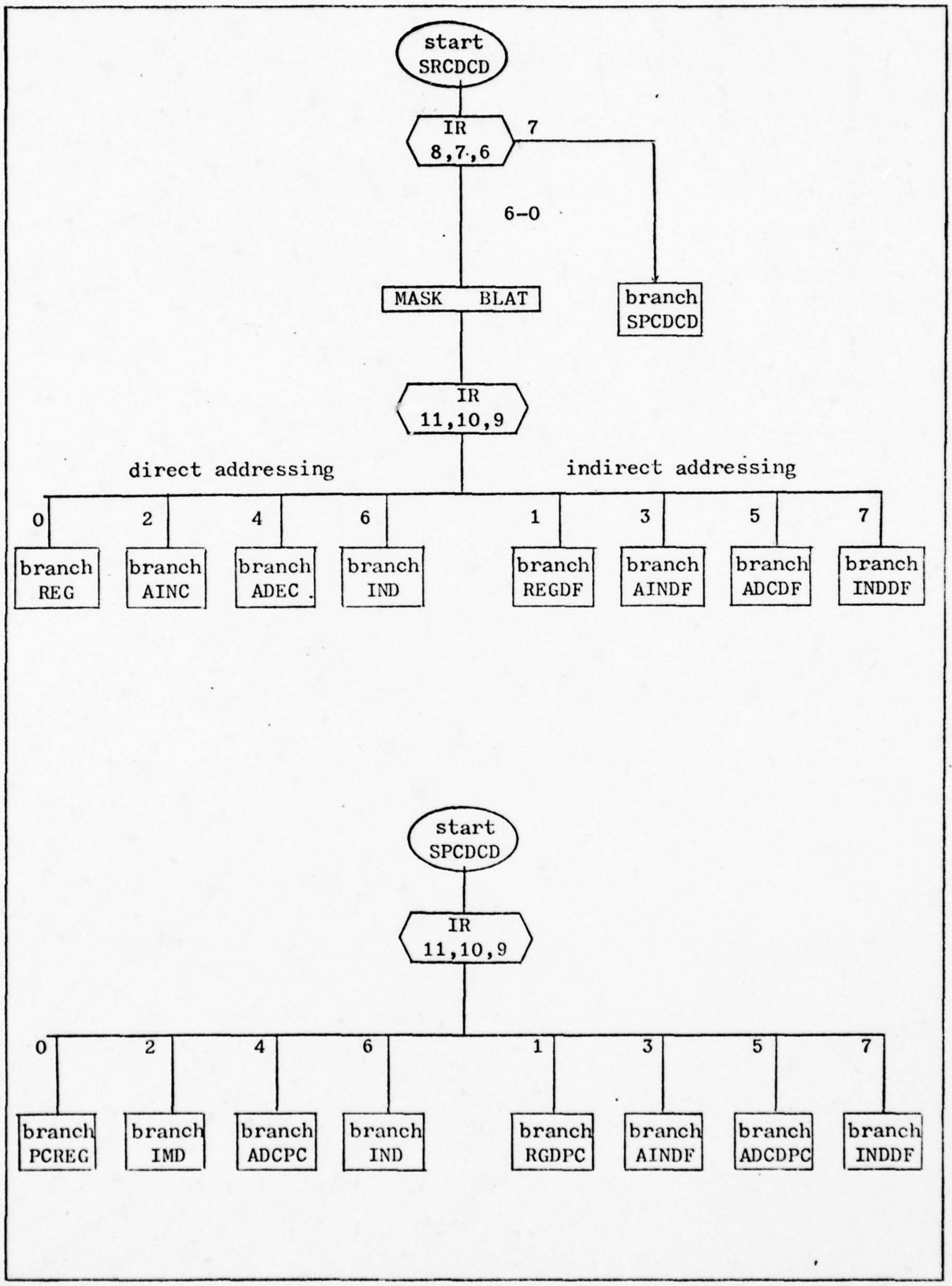


Figure A27. SRCDCD/SPCDCD Flowchart

in Figure A28, MOVB differs in several respects. First, memory reference sources and destinations require testing to determine the pertinent byte. Second, the source byte must be aligned with the destination byte using SWABQ (Figure A29) if they are not both upper or lower. In addition, only a single byte of memory is written into for memory reference destinations. Finally, MOVB requires sign extension through the upper byte for register destinations.

BGE is a conditional branch if greater than or equal to zero which adds two times the offset (sign extended) to the PC if condition codes N and V are both set or clear. The BGE module shown in Figure A30 begins by testing condition codes N and V. If they are not both set or clear, the module branches back to IFTCH. Otherwise, the IR is multiplied by 2 and loaded into the lower byte of ALU general purpose register 13, R13 while the PC is loaded into the DBR. Based on IR07, the lower byte of R13 is sign extended through the upper byte R13 is then added to the DBR which contains the PC and the result loaded back into the PC. Micro branching back to IFTCH allows the macro branch to occur when the new PC is used to fetch the next instruction, assuming no interrupt.

RTS, return from subroutine, loads the contents of the specified register into the PC and pops the memory stack into the register. Referring to Figure A31, the RTS module first loads R into the PC and then calls micro subroutine POP in Figure A32 to retreive the top element of the stack. This is then loaded into the register and a branch to IFTCH initiated.

TRAP stacks the current CCR and PC and loads a new CCR and PC from memory locations octal 000036 and 000034 respectively. As shown in

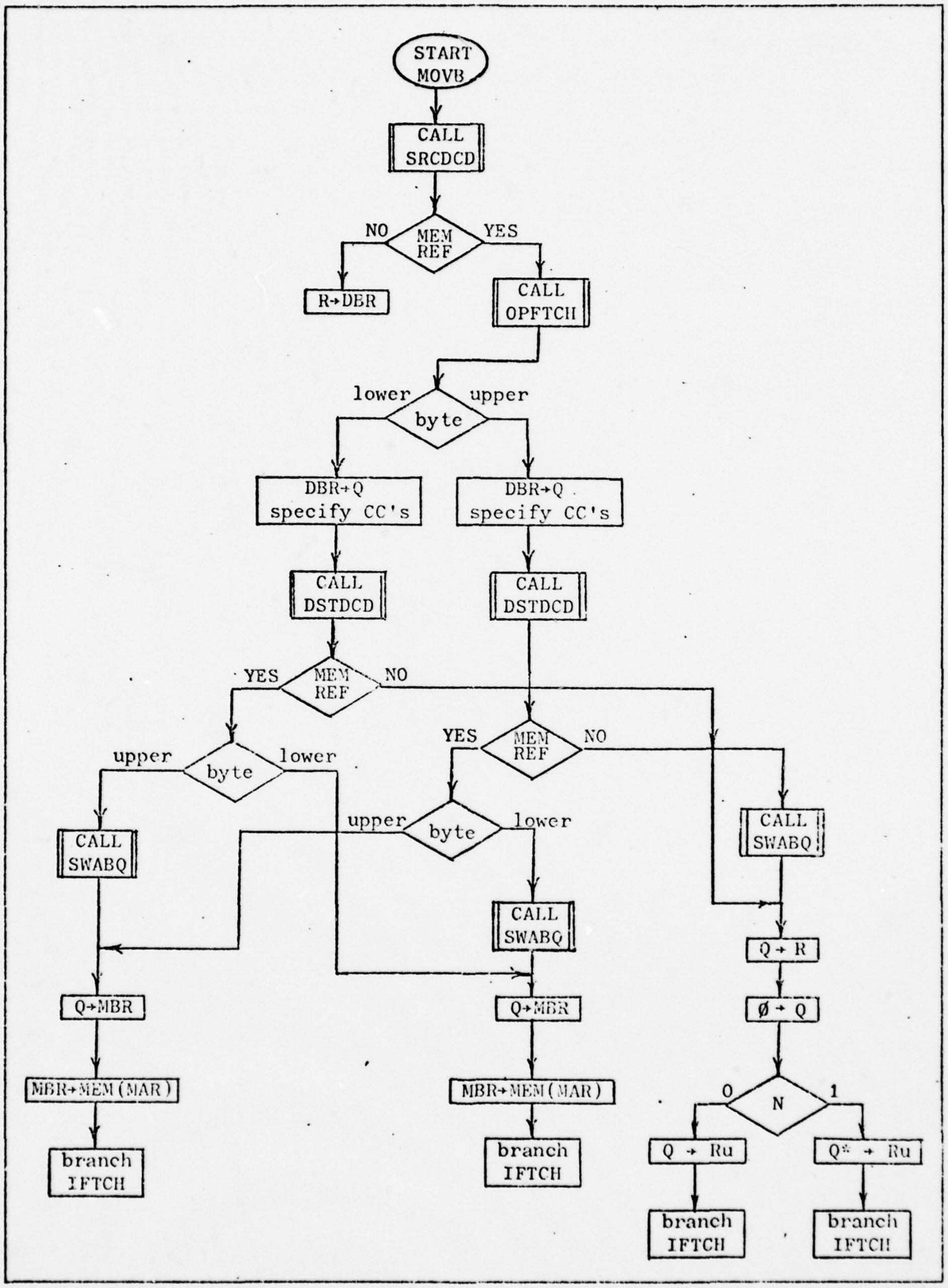


Figure A28. MOVB Flowchart

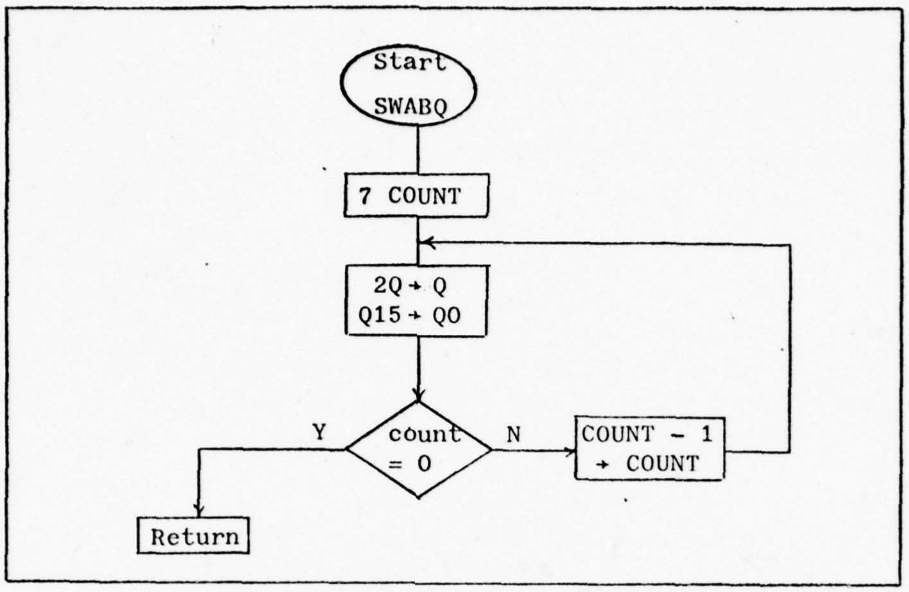


Figure A29. SWABQ Flowchart

Figure A33, this is accomplished by twice loading the MBR and calling PUSH and then loading the PC and CCR with the required constants. As will be discussed further in Appendix A, these constants are stored as part of the microword. Table AXII contains sample microcode generated to implement the emulated PDP-11/03 instructions discussed above.

Summary

The MIME microword was defined in the initial part of this section. Sample microword formats were then presented using various combinations of formatted fields. These two concepts were then illustrated by using microprogramming to emulate several PDP-11/03 instructions in the MIME architecture.

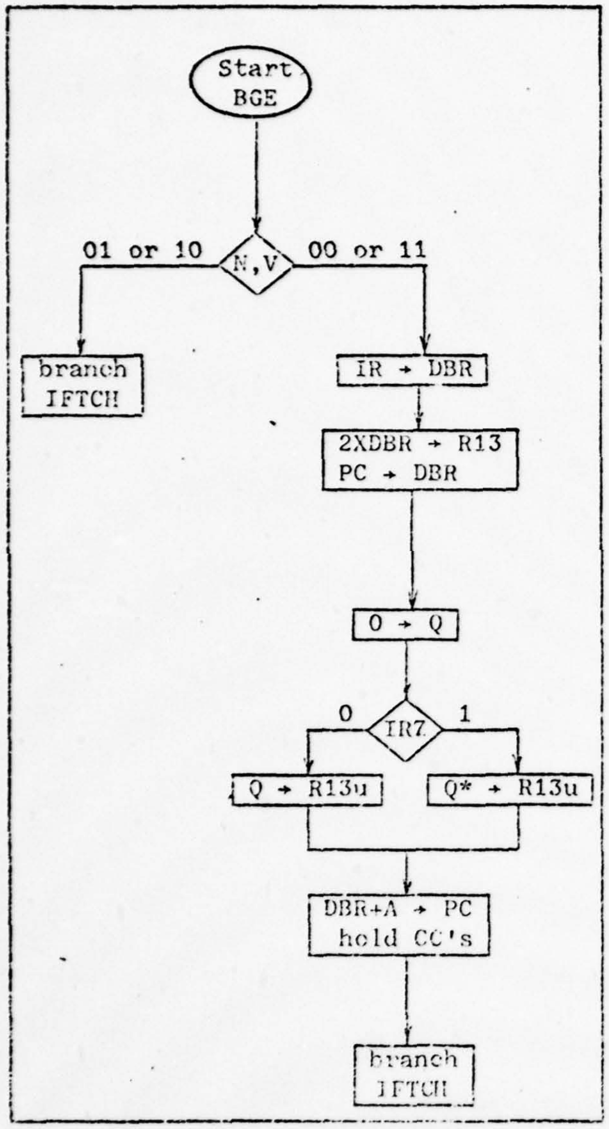


Figure A30. BGE Flowchart

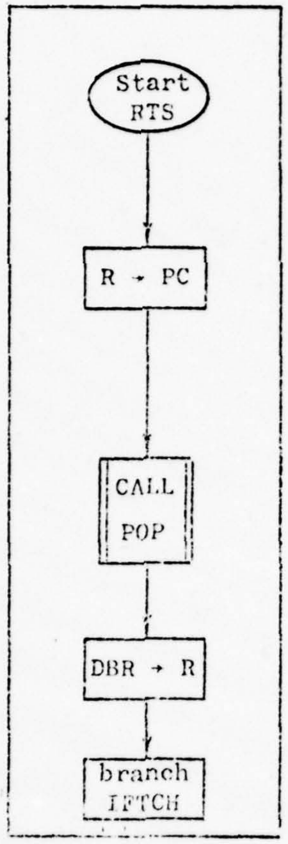


Figure A31.
RTS Flowchart

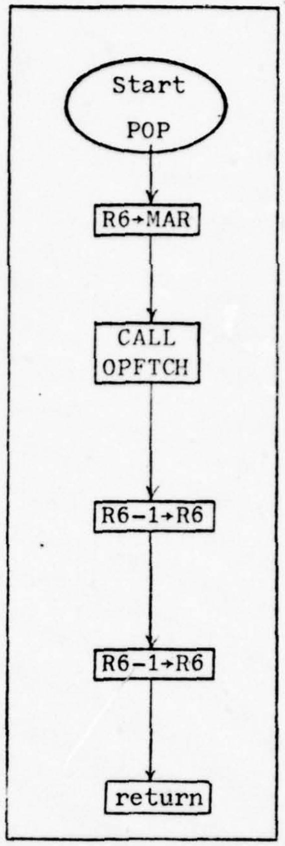


Figure A32.
POP Flowchart

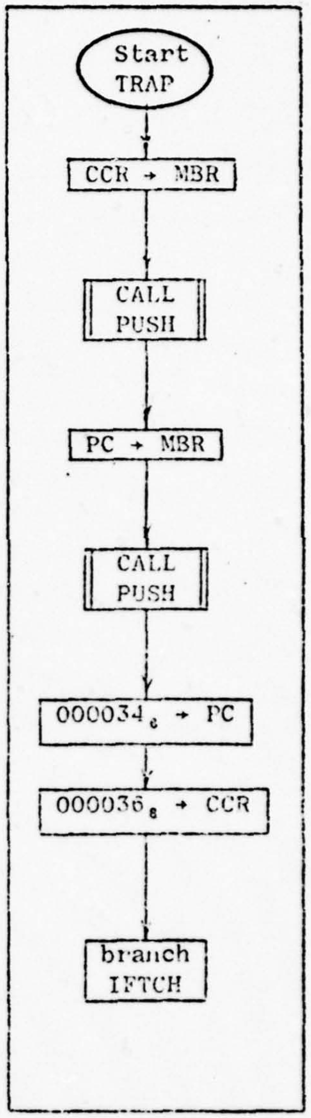


Figure A33.
TRAP Flowchart

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

Table AXII

Sample Microcode

		CONSTANT																			
			BR MUX	BR CONTR	I/O TC MUX		MICRO LOOP COUNTER LOAD														
			POL	CCU TC MUX	ALU TC MUX		MICRO BRANCH ADDR									I/O					
CR ADDR	COMMENTS	N MUX	V MUX	Z MUX	CARRY MUX	CIN MUX	Q MUX	R MUX	CEN MUX	CC*	ALU DST	UB*	ALU FN	LB*	ALU SRC	ALU A ADDR	ALU B ADDR	COM	BUS SRC	BUS DST	MADDR SEQ
		63-62	61-60	59 - 56	55 - 52	51-48	47-44	43-40	39-36	35	34-32	31	30-28	27	26-24	23-20	19-16	15-12	11-8	7-4	3-0
5ØØ	CALL DSTDCD address 789	POS Ø		SET F	X X	X X	7 7	8 8	9 9		MCC,OF 9		X X		X X	X X	X X	NOP F	NOP F	NOP F	CJS 1
5Ø1	Mem Ref Test	NEG 1		MEM 5	X X	X X	5 5	Ø Ø	5 5		MCC,OF 9		X X		X X	X X	X X	NOP F	NOP F	NOP F	CJS 3
5Ø2	CALL OPFTCH address 123	POS Ø		SET F	X X	X X	1 1	2 2	3 3		MCC,OF 9		X X		X X	X X	X X	NOP F	NOP F	NOP F	CJS 1
5Ø3	DBR*←MBR,N:NRM Z:WRD,V:CLR, C:SET	NRM,CLR 6		WRD 5	SET F	X X	X X	X X	X X		CC,OF 1		ON,XNOR 7		ON,DØ 7	X X	X X	NOP F	ALUS 6	MBRL 2	CONT E
5Ø4	MBR←MEM(MAR), branch to IFTCH (ØØ3)	X X		X X	X X	X X	Ø Ø	Ø Ø	3 3		MCC,OF 9		ON,X 8		ON,X 8	X X	X X	NOP F	MARS 1	MEMW 5	JP F
5Ø5	Ri*←Ri,N:NRM, Z:WRD,V:CLR, C:SET branch to IFTCH	NRM,CLR 6		WRD 5	SET F	X X	Ø Ø	Ø Ø	3 3		CC,BOF 3		ON,XNOR 7		ON,ØB 3	X X	X X	NOP F	NOF F	NOP F	JP F

VI. Maintenance

Introduction

This section has been included in the User's Manual for two reasons: first to review MIME troubleshooting features, and second to present procedures used in the initial integration and checkout of MIME.

Troubleshooting

Two features of MIME are particularly useful for troubleshooting: the hardwired front panel and the testpoint array. This section discusses the application of each of these features.

The front panel provides the capability to load and display the contents of registers within MIME and to single step through both macro and micro level programs. Once the user has ascertained that a problem is not caused by front panel malfunction, it may be used for further troubleshooting. Front panel capabilities allow the user to monitor information transfers occurring during program execution which aids isolation of malfunctions to particular circuit cards. Troubleshooting may then be continued at the card level using the test point array discussed below.

The testpoint array shown in Table AXIII is useful for monitoring critical signals within MIME, and in some instances tying these signals high or low for test purposes. The Module Test section contains examples of the use of this technique. If the malfunction cannot be isolated further using the testpoint array, more traditional scope and probe or firmware diagnostic techniques may have to be used.

Table AXIII

Testpoint Array

Testpoint	Function
TP1	+5 volts
TP2	GND
TP3	-5 volts
TP4	+12 volts
TP5	GND
TP6	-12 volts
TP7	STOP*
TP8	WAITRQ*
TP9	FETCH
TP10	MB3 (SYNCH)
TP11	CP1
TP12	CP2
TP13	CP3

Module Tests

In order to limit initial troubleshooting efforts to manageable proportions, MIME was powered-up and checked out incrementally. The front panel was brought up first so that it could be used in checking out the remaining modules. The CCU module was checked out next, providing microinstruction sequencing and decoding. The CS module was then integrated to provide microprogram storage for use with the CCU. The ALU was subsequently checked out by single stepping through microprograms stored in CS. Finally, the I/O module was checked out.

The following sections outline the operational checks performed by the authors to verify correct operation of each particular MIME module. Exhaustive checks of Memory, Control Store, and ALU were not practicable during initial integration because of the time required to do so manually. The authors recognized that diagnostic routines would have been helpful, so their generation was included as a recommendation in the MIME thesis.

Module tests are described using two types of tables. The first type as shown in Table AXIV lists required test equipment and describes set up of MIME cards, cables, and switches. The second type lists the actual test steps. As shown in Table AXV, the function column contains the function being tested, while the ACTION column lists user actions. Front panel switch activations are indicated by capitialized switch names. The OBSERVATION column includes front panel lights and displays, as well as some switching waveforms. The symbol X is used to specify "don't care". The symbols H and L refer to high (logic 1) and low (logic Ø) TTL levels respectively (Ref 7:3-8). The lower case symbols u and l refer to upper and lower byte of the subscripted register. Timing diagram tolerances are ± 50ns unless noted otherwise.

Front Panel. This series of tests checks cards FP1 and FP2 for proper operation. Table AXIV depicts equipment and setup requirements. Table AXV presents test procedures.

Table AXIV

FP Test Equipment
and MIME Setup

Equipment:	
1.	Storage Oscilloscope (Tektronix 464 or equivalent)
2.	IC test clip (AP Products TC-16 or equivalent)
Setup:	
1.	Remove all cards except FP2 (card B)
2.	Check that cables 2 and 4 are connected
3.	Select OCTAL macro display format
4.	Remove red plexiglass front panel cover
5.	Tie Signal STOP* (TP7) to GND
6.	Select NORM micro field displays

Table AXV

FP Test Procedures

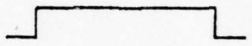
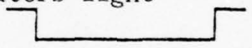
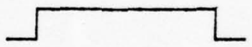
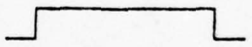
FUNCTION	ACTION	OBSERVATION
Control Logic Test	- Turn power on	- Following indicators will light: - all MACRO data displays - 3 least significant MICRO data displays - PC indicator - A indicator - HLT indicator
	- RUN	- GOREQ (U10-13) 
	- STOP* - VCC - HALT	- N/A - HALTREQ (U3-7) goes high
	- STOP* - GND - RESET	- HALTREQ (U3-7) goes low - PC and A indicators light - RESET* (U13-6) 
	- PAUSE	- STOPREQ (U10-5) 
	- STEP	- SSREQ (U11-13) 
	- AUTO STEP	- ∿1Hz SSREQ (U11-13) pulse until RESET depressed
	- MSTEP	- MSTEPSEL (U3-4) goes high until RESET - MSTEP (U2-6) momentarily high
	- AUTO MSTEP	- AUTOSEL (U3-13) goes high until RESET
Macro Display Logic Test	- Macro >	- Macro select indicator moves one position each time switch is released. Two switch activations are required between Q and RØ.
	- BP1 on	- BP1 (PC305) - H
	- BP2 on	- BP2 (PC306) - H
	- MACRO DISP	- Macro display: 177777. - DISP (U14-11) - H momentarily (except when DB, AB, or RØ-BP2 selected)

Table AXV (Continued)

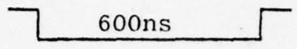
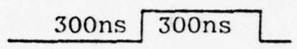
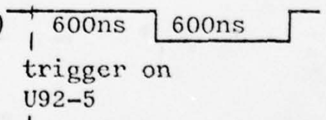
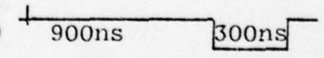
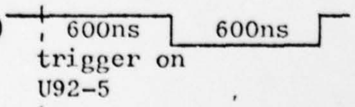
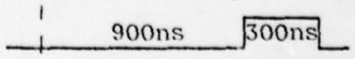
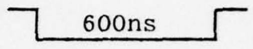
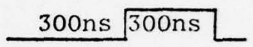
FUNCTION	ACTION	OBSERVATION
	- MACRO DISP NEXT	- same as above, plus if MBR is selected: INCMAR1* (U94-1)  PP3 (U94-5) 
Data Entry Logic Test	- MACRO	- N/A
	- KEYBOARD	- corresponding octal digits enter MACRO display (most significant display 1 or Ø only)
	- HEX	- 2 most significant MACRO displays blanked
	- KEYBOARD	- corresponding hex digits enter MACRO display
	- MICRO	- N/A
	- KEYBOARD	- corresponding hex digits enter MICRO display
	- FLDEN(S)	- N/A
	- KEYBOARD	- Hex digits enter enabled MICRO displays
	- LOAD	- MIDD* (U96-6)  - MPP1* (U96-8) 
	- LOAD NEXT	- same as above, plus if CS is selected INCMA (U66-1) - H momentarily
	- MACRO - LOAD	- N/A - LDD* (U96-3)  - PP1 (U9-12) 

Table AXV (Continued)

FUNCTION	ACTION	OBSERVATION
	- LOAD NEXT	- same as above, plus if MBR is selected: INCMAR1* (U94-1)  PP3 (U94-5) 
Micro Display Logic Test	- MICRO >	- Micro select indicator moves one position each time switch depressed. 13 most significant MICRO displays blanked when in A or BP.
	- BP on	- MBP (PC501) - H
	- DISP	- Micro display: FFFF,FFFF,FFFF,FFFF
	- DISP NEXT	- same as above, plus if CS is selected: - INCMA (U66-1) - H momentarily

CCU. This series of tests checks cards CCU1 and CCU2 for proper operation. Equipment and set requirements are presented in Table AXVI, test procedures in Table AXVII. Mnemonics such as JZ in the FUNCTION column of Table AXXVIII refer to the particular microaddress sequencer instruction being tested. These same mnemonics are used in the OBSERVATION field beside their associated microinstructions. T = H or L refers to the test condition value associated with conditional sequencer instructions. Quantities in parenthesis such as (ABB→PC) refer to internal sequencer transfers as explained in Reference 2.

Table AXVI

CCU Test Equipment and Setup

Equipment:	
1.	Multimeter (Triplett 63Ø-NS or equivalent)
2.	Logic Clip (HP 1Ø528A or equivalent)
3.	Extender card
Setup:	
1.	Remove all cards except FP1, FP2(B), CCU1(D), and CCU2(E)
2.	Place CCU1 on extender
3.	Check that cables 2, 3, 4, and 5 are connected
4.	Select NORM micro field displays
5.	CCU:S1-OFF, S2-OFF, S3-OFF, S4-OFF

Table AXVII

CCU Tests

FUNCTION	ACTION	OBSERVATION
PL Test	- Turn on Power (+5)	- HLT on, PC on, A on
	- MSTEP	- N/A (change clock mode to manual)
	- DISP PL	- Micro display: FFFF,FFFF,FFFF,FFF7$_{16}$
	- LOAD PL	- Micro display: test data (user specified)
	- change micro display	- Micro display: X
	- DISP PL	- Micro display: test data
PC Test	- DISP PC	- Macro display: ØØØØ$_{16}$
	- LOAD PC	- Macro display: test data
	- change macro display	- Macro display: X
	- DISP PC	- Macro display: test data
	- LOAD PL	- Micro display: XXXX,XXXX,XXXX,ØFFX
	- RUN, MSTEP, MSTEP	- RUN on
	- DISP PC	- Macro display: test data +1
IR Test	- LOAD IR	- Macro display: test data
	- Change Macro display	- Macro display: X
	- DISP IR	- Macro display: test data
Initial Starting Addr Test	- DISP A	- Micro display: CØØ
	- S2 - ON, RESET	- N/A
	- DISP A	- Micro display: 4ØØ
	-S3 - ON, RESET	- N/A
	- DISP A	- Micro display: ØØØ
Decode Test (without mapping PROM)	- S4 - ON	- N/A
	- LOAD IR	- Macro display: AA55
	- LOAD PL	- Micro display: XXXX,XXXX,XXXX,AFF2
		- TP5(WAITRQ*) ¯¯|_450ns_|¯¯
	- DISP A	- Micro display: 255 (IR 9-Ø)
	- LOAD PL	- Micro display: XXXX,XXXX,XXXX,ØXX2
	- DISP A	- Micro display: 2A9(IR15-6)

Table AXVII (Continued)

FUNCTION	ACTION	OBSERVATION
Sequencer Tests JZ	- LOAD PL	- Micro display: XXXX,XXXX,XXXX,FFFØ (JZ)
	- DISP A	- Micro display: ØØØ
CJS, CRTN	- RESET	- N/A
	- LOAD PL	- Micro display: OFXX,AAAX,XXXX,FFF1 (CJS,T=H)
	- DISP A	- Micro display: AAA
	- RESET (AAB→PC)	- N/A
	- LOAD PL	- Micro display: IFXX,BBBX,XXXX,FFF1 (CJS,T=L)
	- DISP A	- Micro display: AAB (AAA+1)
	- LOAD PL	- Micro display: IFXX,XXXX,XXXX,FFFA (CRTN,T=L)
	- DISP A	- Micro display:AAB
	- LOAD PL	- Micro display: OFXX,XXXX,XXXX,FFFA (CRTN,T=H)
	- DISP A	- Micro display: 001
JMAP	- LOAD IR	- Macro display: AA55
	- LOAD PL	- Micro display: XXXX,XXXX,XXXX,FFF2 (JMAP)
	- DISP A	- Micro display: 2A9 (IR15-6)
CJP	- LOAD PL	- Micro display: OFXX,AAAX,XXXX,FFF3 (CJP,T=H)
	- DISP A	- Micro display: AAA
	- RESET (AAB→PC)	- N/A
	- LOAD PL	- Micro display: 1FXX,BBBX,XXXX,FFF3 (CJP,T=L)
	- DISP A	- Micro display: AAB (AAA+1)
PUSH RFCT	- LOAD PL	- Micro display: OFXX,001X,XXXX,FFF4 (PUSH,T=H)
	- DISP A	- Micro display: AAB
	- RESET (AAC→F)	- N/A
	- LOAD PL	- Micro display: 16XX,XXXX,XXXX,FFF8 (RFCT, count≠0)
	- DISP A	- Micro display: AAC (file)
	- RESET (AAD→PC)	- N/A
	- LOAD PL	- Micro display: 16XX,XXXX,XXXX,FFF8 (RFCT, count=0)
	- DISP A	- Micro display: AAD (AAC+1)

Table AXVII (Continued)

FUNCTION	ACTION	OBSERVATION
JP,JRP	- LOAD PL	- Micro display: XXXX,555X,XXXX,FFFF (JP)
	- DISP A	- Micro display: 555
	- RESET (555→R)	- N/A
	- LOAD PL	- Micro display: 1FXX,AAAX,XXXX,FFF7 (JRP,T=L)
	- DISP A	- Micro display: 555
	- LOAD PL	- Micro display: 0FXX,AAAX,XXXX,FFF7 (JRP,T=H)
	- DISP A	- Micro display: AAA
CONT, JSRP	- LOAD PL	- Micro display: XXXX,666X,XXXX,FFFF (JP)
	- RESET (667→PC)	- N/A
	- LOAD PL	- Micro display: XXXX,777X,XXXX,FFFE (CONT)
	- DISP A	- Micro display: 667
	- RESET (668→PC,777→R)	- N/A
	- LOAD PL	- Micro display: 1FXX,BBBX,XXXX,FFF5 (JSRP,T=L)
	- DISP A	- Micro display: 777
	- RESET (PUSH 668)	- N/A
	- LOAD PL	- Micro display: 0FXX,XXXX,XXXX,FFFA (CRTN,T=H)
	- DISP A	- Micro display: 668
JSRP	- LOAD PL	- Micro display: XXXX,777X,XXXX,FFFE (CONT)
	- RESET (778→PC)	- N/A
	- LOAD PL	- Micro display: 0FXX,CCCX,XXXX,FFF5 (JSRP,T=H)
	- DISP A	- Micro display: CCC
	- RESET (PUSH 778)	- N/A
	- LOAD PL	- Micro display: 0FXX,XXXX,XXXX,FFFA (CRTN,T=H)
	- DISP A	- Micro display: 778
LDCT, RPCT	- LOAD PL	- Micro display: XXXX,002X,XXXX,FFFC (LDCT)
	- DISP A	- Micro display: 778
	- RESET (002→CT)	- N/A
	- LOAD PL	- Micro display: 16XX,AAAX,XXXX,FFF9 (RPCT count≠0)
	- DISP A	- Micro display: AAA
	- RESET (DEC CT,AAB→PC)	- N/A
	- LOAD PL	- Micro display: 16XX,AAAX,XXXX,FFF9 (RPCT count≠0)
	- DISP A	- Micro display: AAA

Table AXVII (Continued)

FUNCTION	ACTION	OBSERVATION
LDCT, RPCT (cont)	- RESET (DEC CT, AAB→PC) - LOAD PL - DISP A	- N/A - Micro display: 16XX,AAAX,XXXX,FFF9 (RPCT count=0) - Micro display: AAB (AAA+1)
LOAD A	- S4 - OFF - LOAD A - DISP A	- N/A - Micro display: 006 - Micro display: 006
CJPP, LOOP	- RESET (007→PC) - S4 - ON - LOAD PL - RESET (PUSH 007) - LOAD PL - RESET (FOD→PC) - LOAD PL - RESET (PUSH FOD, FOE→PC) - LOAD PL - DISP A - LOAD PL - DISP A - RESET (POP FOD) - LOAD PL - DISP A - RESET (008→PC) - LOAD PL - DISP A	- N/A - N/A - Micro display: XXXX,XXXX,XXXX,FFF4 (PUSH) - N/A - Micro display: XXXX,FOCX,XXXX,FFFF (JP) - Micro display: XXXX,XXXX,XXXX,FFF4 (PUSH) - N/A - Micro display: OFXX,777X,XXXX,FFFB (CJPP,T=H) - Micro display: FOE - Micro display: 1FXX,777X,XXXX,FFFB (CJPP,T=L) - Micro display: 777 - N/A - Micro display: 1FXX,XXXX,XXXX,FFFD (LOOP,T=H) - Micro display: 007 - N/A - Micro display: OFXX,XXXX,XXXX,FFFD (LOOP,T=L) - Micro display: 008
Microbreak	- S4 - OFF - LOAD BP (micro)	- N/A - Micro display: ABC

Table AXVII (Continued)

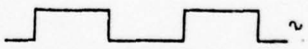
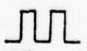
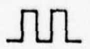
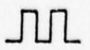
FUNCTION	ACTION	OBSERVATION
Microbreak (cont)	- DISP BP	- Micro display: ABC - MB3 (TP1Ø) - L
	- LOAD A	- Micro display: ABC - MB3 - H
Clock Logic	- RESET	- N/A
	- MSTEP	- N/A
	- RUN	- N/A
	- MSTEP	- RUN on ; - CP1 (TP11) - H momentarily
	- PAUSE (hold)	- N/A
	- MSTEP, release PAUSE	- HLT on
	- MSTEP	- N/A
	- RUN, MSTEP	- RUN on
	- WAITRQ* (TP8) - GND	- N/A
	- MSTEP	- CP1 - H momentarily
	- MSTEP	- CP1 - L
	- TP5 - REMOVE GND	- N/A
	- MSTEP	- CP1 - L
	- MSTEP	- CP1 - H momentarily
SS Logic (including front panel)	- RESET	- N/A
	- FETCH (TP9) - GND	- N/A
	- AUTO MSTEP	- CP1  ∿ 1.5Hz
	- RESET	- N/A
	- RUN	- CP1  7ØØns cycle (±35ns) - RUN on
	- FETCH - remove GND	- N/A
	- HALT	- CP1 - L ; HLT on
	- FETCH - GND	- N/A
	- AUTO STEP	- CP1 -  ∿7ØØns cycle (±35ns) - RUN on
	- FETCH - remove GND	- CP1 - L ; HLT on
	- FETCH - GND	- CP1 -  ∿7ØØns cycle (±35ns) - RUN on
	- FETCH - remove GND	- HLT on
	- RESET	- N/A

Table AXVII (continued)

FUNCTION	ACTION	OBSERVATION
16 Way Branch	- S4 - ON - LOAD IR - LOAD PL	- N/A - Macro display: FFFØ$_{16}$ - Micro display: ØFXX,AAØX,XXXX,5FFF (JPL)
	- DISP A	- Micro display: AAØ
	- LOAD IR	- Macro display: FFFA
	- DISP A	- Micro display: AAA
	- LOAD PL	- Micro display: ØAXX,AAØX,XXXX,5FFF (JPL)
	- DISP A	- Micro display: AA2
	- LOAD PL	- Micro display: ØAXX,AAØX,XXXX,FFFF (JPL)
	- DISP.A	- Micro display: AAØ

CS1. These tests verify proper operation of the RAM portion of CS1. EPROM operation may be verified using similar procedures. Table AXVIII lists equipment requirements and MIME setup procedures. Test procedures are presented in Table AXIX. Quantities in parentheses beside CS contents are the address of that microword.

Table AXVIII

CS1 Test Equipment
and MIME Setup

Equipment:	
1.	Storage oscilloscope (Tektronix 464 or equivalent)
2.	Logic clip (AP Products TC-16 or equivalent)
Setup:	
1.	Remove ALU, I/O, and MEM1 cards
2.	Place CS1 card on extender
3.	Check that cables 1, 2, 3, 4, 5, and 7 are connected
4.	Select NORM micro field displays
5.	CCU2: S1 - OFF, S2 - ON, S3 - ON, S4 - OFF
6.	Select MICRO data entry

Table AXIX

CS1 Tests

FUNCTION	ACTION	OBSERVATION
Front Panel LOAD and DISP	- Turn on Power (+5v)	- HLT on, PC on, A on
	- LOAD A	- Micro display: ØXX (test addr., Ø to 253_{10})
	- LOAD NEXT CS	- Micro display: AAAA,5555,AAAA,55555 (ØXX)
	- LOAD NEXT CS	- Micro display: 5555,AAAA,5555,AAAA (ØXX+1)
	- DISP A	- Micro display: ØXX+2
	- LOAD A	- Micro display: ØXX
	- DISP NEXT CS	- Micro display: AAAA,5555,AAAA,5555 (ØXX)
	- DISP CS	- Micro display: 5555,AAAA,5555,AAAA (ØXX+1)
	- DISP A	- Micro display: 0XX+1
	- LOAD A	- Micro display: 4YY (test addr)
	- LOAD CS	- Micro display: 1111,0000,1111,0000 (4YY)
	- DISP CS	- Micro display: FFFF,FFFF,FFFF,FFFF (4YY)
Front Panel FLDEN	- LOAD A	- Micro display: 0XX
	- FLDEN (least sig.)	- N/A
	- LOAD CS	- Micro display: N/A (trigger MLD* - CU38-05, U14) - E4*(CU37-12,R16) ∿900ns 300ns (±50ns) - E3*(CU36-6,P13) ∿900ns 300ns (±50ns) - E2*(CU36-8,N12) ∿900ns 300ns - E1*(CU36-12,N16)

Table AXIX (Continued)

FUNCTION	ACTION	OBSERVATION
Front Panel FLDEN (cont)	- FLDEN (2nd least sig.)	- N/A
	- LOAD CS	- Micro display: N/A
		- E4* ~900ns $\boxed{300ns}$____ (±50ns)
		- E3* ~900ns $\boxed{300ns}$____ (±50ns)
		- E2* ____________
		- E1* ____________
	- FLDEN(3rd least sig.)	- N/A
	- LOAD CS	- Micro display: N/A
		- E4* ~900ns $\boxed{300ns}$____ (±50ns)
		- E3* ____________
		- E2* ____________
		- E1* ____________
	- FLDEN (most sig.)	- N/A
	- LOAD CS	- Micro display: N/A
		- E4* ____________
		- E3* ____________
		- E2* ____________
		- E1* ____________
EPROM Protect, asynch. logic	- NORM (all fields)	- N/A
	- LOAD A	- Micro display: 4XX (an address in EPROM)
	- LOAD CS	- Micro display: N/A (trigger MLD* - CU38-05,U14)
		- E4* ~900ns $\boxed{300ns}$____
		- E3* ~900ns $\boxed{300ns}$____
		- E2* ~900ns $\boxed{300ns}$____
		- E1* ~900ns $\boxed{300ns}$____
	- LOAD A	- Micro display: ØXX (an address in RAM)
		- WAITRQ*(TP9) ~250ns $\left(\begin{smallmatrix}+50ns,\\-0ns\end{smallmatrix}\right)$
	- LOAD A	- Micro display: 4XX
		- WAITRQ* ~450ns $\left(\begin{smallmatrix}+50ns,\\-0ns\end{smallmatrix}\right)$

ALU. This series of tests checks ALU operation. Equipment requirements and setup procedures are presented in Table AXX, and test procedures in Table AXXI.

Table AXX

ALU Test Equipment
and MIME Setup

Equipment:
- Extender card
Setup:
1. Remove I/O and MEM1 cards 2. Place ALU card on extender 3. Check that all cables are connected 4. Select NORM micro field display 5. CCU2: S1 - OFF, S2 - ON, S3 - ON, S4 - OFF

Table AXXI

ALU Tests

FUNCTION	ACTION	OBSERVATION
Front Panel Interface	- Turn on Power (+5v)	- HLT on, PC on, A on
	- LOAD ALU	- Macro display: test data A
	- LOAD Q	- N/A
	- Change macro display	- Macro display: -
	- DISP Q	- Macro display: test data A
	- LOAD ALU	- Macro display: test data B
	- LOAD R9	- N/A
	- Change macro display	- Macro display: -
	- DISP R9	- Macro display: test data B
	- DISP CCR	- Macro display: $XXØØ_{16}$
	- LOAD CCR	- Macro display: test data C
	- Change macro display	- N/A
	- DISP CCR	- Macro display: test data C
MSKB Test	- LOAD PL	- Micro display: AA55,XXXX,XXXX,FDFF
	- DISP DB	- Macro display: AA55
	- RESET	- N/A
	- DISP DB	- Macro display: FFFF
Arithmetic Tests micro-program	- RESET	- N/A
	- LOAD ALU	- Macro display: ØØ55
	- LOAD R5	- Macro display: ØØ55
	- LOAD ALU	- Macro display: AAØØ
	- LOAD R1Ø	- Macro display: AAØØ
	- LOAD A	- Micro display: Ø11 (initial microaddress)
(QOF,ADD,AB)	- LOAD NEXT CS	- Micro display: E7E6,XXXØ,Ø15A,8F6E (Ø11)
(BOF,S-R,AQ)	- LOAD NEXT CS	- Micro display: B6F1,XXX3,1Ø52,8F6E (Ø12)
	- RESET	- N/A
	- LOAD A	- Micro display: Ø11

Table AXXI (Continued)

FUNCTION	ACTION	OBSERVATION
Arithmetic Tests Micro-Program (cont)	- Select DB display	- DB on
	- MSTEP, RUN, MSTEP	- Micro display: Ø11 ; RUN on - Macro display: FFFF, N=Ø,Z=Ø,C=Ø,OVR=Ø
	- MSTEP	- Micro display: Ø12 - Macro display: AA55,N=1,Z=1,C=Ø,OVR=Ø
	- MSTEP	- Micro display: Ø13 - Macro display: AAØØ, N=0,Z=0,C=1,OVR=1
	- RESET	- HLT on
	- DISP R2	- Macro display: AAØØ
	- DISP R5	- Macro display: ØØ55
	- DISP Q	- Macro display: AA55
Logic Test Micro-Program	- LOAD ALU	- Macro display: AAAA
	- LOAD R7	- Macro display: AAAA
	- LOAD ALU	- Macro display: 5555
	- LOAD A	- Micro display: ØAØ
(QOF,AND,DA)	- LOAD NEXT CS	- Micro display: XXXX,XXXØ,457X,8FFE (ØAØ)
(QOF,OR,DA)	- LOAD NEXT CS	- Micro display: XXXX,XXXØ,357X,8FFE (ØA1)
	- RESET	- N/A
	- LOAD A	- Micro display: ØAØ
	- Select ALU display	- ALU on
	- MSTEP, RUN, MSTEP	- Micro display: ØAØ ; RUN on - Macro display: FFFF
	- MSTEP	- Micro display: ØA1 - Macro display: ØØØØ
	- MSTEP	- Micro display: ØA2 - Macro display: FFFF
	- RESET	- HLT on

Table AXXI (Continued)

FUNCTION	ACTION	OBSERVATION
Shift/ Byte tests	- LOAD ALU	- Macro display: 781B
	- LOAD R1	- Macro display: 781B
	- LOAD ALU	- Macro display: 5555
	- LOAD Q	- Macro display: AAAA
	- LOAD CCR	- Macro display: XXX8
	- LOAD A	- Micro display: ØBC
rotate word R and Q right, carry	- LOAD NEXT CS	- Micro display: XXXX,223C,37ØF,8FFE (ØBC)
rotate word R and Q left, one	- LOAD NEXT CS	- Micro display: XXXX,443E,37ØE,8FFE (ØBD)
rotate lower byte right	- LOAD NEXT CS	- Micro display: XXXX,XXØD,B3Ø1,8FFE (ØBE)
rotate upper byte left	- LOAD NEXT CS	- Micro display: XXXX,XXØF,3BØØ,FFFE (ØBF)
	- RESET	- N/A
	- LOAD A	- Micro display: (ØBC)
	- Select ALU display	- ALU on
	- MSTEP, RUN, MSTEP	- Micro display: ØBC ; RUN on - Macro display: FFFF
	- MSTEP	- Micro display: ØBD - Macro display: 5555
	- MSTEP	- Micro display: ØBE - Macro display: AAAA
	- MSTEP	- Micro display: ØBF - Macro display: 781B
	- MSTEP	- Micro display: ØCØ - Macro display: 78ØD
	- RESET	- HLT on
	- DISP R1	- Macro display: FØØD
	- DISP R14	- Macro display: 5555
	- DISP R15	- Macro display: AAAA
	- DISP Q	- Macro display: 5555

MEM 1. These tests verify proper operation of the MEM 1 card. Equipment requirements and MIME set up procedures are presented in Table AXXII. Table AXXIII contains the test procedures.

Table AXXII

MEM 1 Test Equipment and MIME Setup

Equipment:
- Extender card
Setup:
1. Remove I/O card 2. Place MEM1 card on extender 3. Check that all cables are connected 4. Select NORM micro field display 5. CCU: S1 - OFF, S2 - ON, S3 - ON, S4 - OFF

Table AXXIII

MEM1 Tests

FUNCTION	ACTION	OBSERVATION
Front Panel LOAD and DISP	- Turn on Power (+5v)	- HLT on, PC on, A on
	- LOAD MAR	- Macro display: ØØA5
	- LOAD MBR	- Macro display: 44BB (ØØA5)
	- DISP MAR	- Macro display ØØA5
	- DISP MBR	- Macro display: 44BB (ØØA5)
	- LOAD MAR	- Macro display: Ø123
	- LOAD NEXT MBR	- Macro display: FEED (Ø123)
	- LOAD NEXT MBR	- Macro display: BABE (Ø124)
	- DISP MAR	- Macro display: Ø125
	- LOAD MAR	- Macro display: Ø123
	- DISP NEXT MBR	- Macro display: FEED (Ø123)
	- DISP NEXT MBR	- Macro display: BABE (Ø124)
Program Control	- LOAD MAR	- Macro display: Ø25A
	- LOAD MBR	- Macro display: ØØØØ
	- LOAD ALU	- Macro display: ØØCD
	- LOAD RØ	- Macro display: ØØCD
	- LOAD ALU	- Macro display: ABØØ
	- LOAD R1	- Macro display: ABØØ
	- LOAD A	- Micro display: Ø77
(RØ→MBR)	- LOAD NEXT CS	- Micro display: XXXX,XXX1,33XØ,762E (Ø77)
(MBR→MEM$_L$)	- LOAD NEXT CS	- Micro display: XXXX,XXX1,8ØXX,F15E (Ø78)
(R1→MBR)	- LOAD NEXT CS	- Micro display: XXXX,XXX1,33X1,762E (Ø79)

Table AXXIII (Continued)

FUNCTION	ACTION	OBSERVATION
(MBR→MEM$_H$)	- LOAD NEXT CS	- Micro display: XXXX,XXX1,Ø8XX,F15E (Ø7A)
	- RESET	- N/A
	- LOAD A	- Micro display: Ø77
	- SELECT DB display	- DB on
	- MSTEP, RUN, MSTEP	- Micro display: Ø77 ; RUN on - Macro display: FFFF
	- MSTEP	- Micro display: Ø78 - Macro display: ØØCD
	- MSTEP	- Micro display: Ø79 - Macro display: FFFF
	- MSTEP	- Micro display: Ø7A - Macro display: ABØØ
	- MSTEP	- Micro display: Ø7B - Macro display: FFFF
	- RESET	- HLT on
	- DISP MBR	- Macro display: ABCD

I/O. This series of tests checks I/O card operation. Table AXXIV contains equipment requirements and MIME setup procedures. Table AXXV contains test procedures.

Table AXXIV

I/O Test Equipment
and MIME Setup

Equipment:
- Extender card
Setup:
1. All cards in place 2. Place I/O card on extender 3. Check that all cables are connected 4. Select NORM micro field display 5. CCU: S1 - OFF, S2 - ON, S3 - ON, S4 - OFF

Table AXXV

I/O Tests

FUNCTION	ACTION	OBSERVATION
Front Panel Integration	- Turn on Power (+5v)	- HLT on, PC on, A on
	- LOAD MR	- Macro display: XXAA
	- LOAD SR	- Macro display: XXX5
	- DISP MR	- Macro display: FFAA
	- DISP SR	- Macro display: FFF5
	- LOAD WCR	- Macro display: 57AB
	- LOAD BAR	- Macro display: BA75
	- DISP WCR	- Macro display: 57AB
	- DISP BAR	- Macro display: BA57
Program Control Operation	- LOAD WCR	- Macro display: FFFE
	- LOAD BAR	- Macro display: FFFD
	- LOAD ALU	- Macro display: XXØ8
	- LOAD Q	- Macro display: XXØ8 (For use as Mask)
	- LOAD ALU	- Macro display: XXX2 (For use as Status)
	- LOAD A	- Micro display: Ø55
MCL	- LOAD NEXT CS	- Micro display: XXXX,XXX9,XXØX,CFFE (Ø55)
LMR	- LOAD NEXT CS	- Micro display: XXXX,XXX9,32EX,CFFE (Ø56)
LSR	- LOAD NEXT CS	- Micro display: XXXX,XXX9,379X,C6FE (Ø57)
continue	- LOAD NEXT CS	- Micro display: XXXX,XXX9,XXXX,FFFE (Ø58)

Table AXXV (Continued)

FUNCTION	ACTION	OBSERVATION
RDV	- LOAD NEXT CS	- Micro display: 57XX,Ø6Ø9,XX5X,CFFF (Ø59)
	- RESET	- N/A
	- LOAD A	- Micro display: Ø64
RSR	- LOAD NEXT CS	- Micro display: XXXX,XXX9,XX6X,C9FE (Ø64)
RMR	- LOAD NEXT CS	- Micro display: XXXX,XXX9,XX7X,C9FE (Ø65)
	- RESET	- N/A
	- LOAD A	- Micro display: Ø55
	- Select DB display	- DB on
	- MSTEP, RUN, MSTEP	- Micro display: Ø55 ; RUN on - Macro display: FFFF
MCL	- MSTEP	- Micro display: Ø56 - Macro display: FFFF
LMR	- MSTEP	- Micro display: Ø57 - Macro display: XXØ8
LSR	- MSTEP	- Micro display: Ø58 - Macro display: XXX2
(int. req. P4)	- Momentarily GND P4 (HU31-26,H36)	- N/A
	- MSTEP	- Micro display: Ø59 - Macro display: FFFF ; INT on
RDV	- MSTEP	- Micro display: Ø64 - Macro display: FFFF
RSR	- MSTEP	- Micro display: Ø65 - Macro display: FFF5
RMR	- MSTEP	- Micro display: Ø66 - Macro display: FFØ8
	- RESET	- HLT on
Bus Contention Logic	- LOAD A	- Micro display: Ø64
	- FLDEN (L.S.)	- N/A
RSR (w/contention)	- LOAD NEXT CS	- Micro display: XXXX,XXXX,XXXX,CFFE
RMR (w/contention)	- LOAD NEXT CS	- Micro display: XXXX,XXXX,XXXX,CFFE
	- RESET	- N/A
	- LOAD A	- Micro display: Ø64
	- NORM (all)	- N/A
	- Select DB display	
	- MSTEP, RUN, MSTEP	- Micro display: Ø64 ; RUN on - Macro display: FFFF

Table AXXV (Continued)

FUNCTION	ACTION	OBSERVATION
RSR	- MSTEP	- Micro display: Ø65 - Macro display: FFFF
RMR	- MSTEP	- Micro display: Ø66 - Macro display: FFFF
	- RESET	- HLT on
	- DISP SR	- Macro display: FFF5
	- DISP MR	- Macro display: FFØ8

List of References

1. Purvis, Richard E. and Ronald D. Yoho. MIME: Microprogrammable Minicomputer Emulator. Unpublished thesis. Wright-Patterson Air Force Base: Air Force Institute of Technology, Mar 78.

2. Mick, John R. and Jim Brick. Microprogramming Handbook and Am2910 Emulation. (Second Edition) Sunnvale: Advanced Micro Devices, 1977.

3. Am2900 Bipolar Microprocessor Family. Product Brochure. Sunnyvale: Advanced Micro Devices, 1976.

4. Am2914 Priority Interrupt Encoder. Product Brochure. Sunnyvale: Advanced Micro Devices, 1976.

5. PDP-11/03 Processor Handbook. Maynard: Digital Equipment Corporation, 1975.

6. Pennett, Gary and Saleem Iftekar. Design of an R232 Interface for the Microprogrammable Minicomputer Emulator. Unpublished laboratory report. Wright-Patterson Air Force Base: Air Force Institute of Technology, 1978.

7. The TTL Data Book. (Second Edition) Dallas: Texas Instruments Inc., 1976.

Appendix 1

Schematics

This appendix contains a schematic that describes the electrical configuration of each circuit card. In order to simplify the drawings, the following conventions were used:

1. Even though the logic gates used were on IC's, individual logic symbols were used for functional clarity.
2. Where two or more IC's of the same type are adjacent, pin numbers for the connections are given for only one, and the others may be assumed to follow the same pattern.
3. A connection between intersecting lines is designated by a dot.
4. A fillet connecting two lines indicates a connection to a bus. Where an IC has several adjacent connections to a bus, only the first and last connections are named.
5. Connections to edge or cable connectors are designated by the symbol -<<.

The user will find it helpful to use the schematics in conjunction with the wiring tables which are available from the laboratory staff. A sketch of the component layout is included for FP1, but the wiring tables contain the information for all other circuit cards.

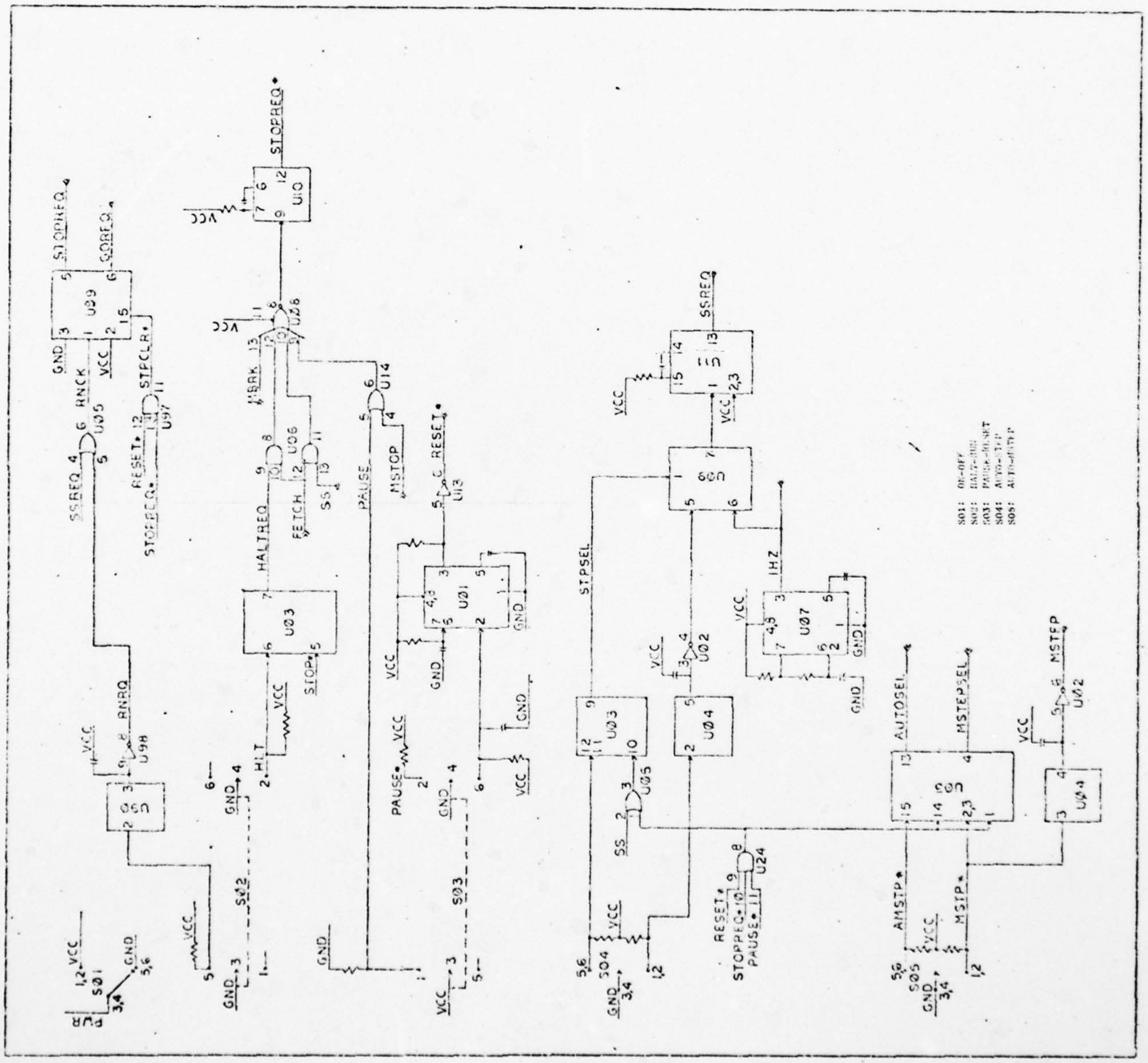


Figure A34. FP1 Schematic

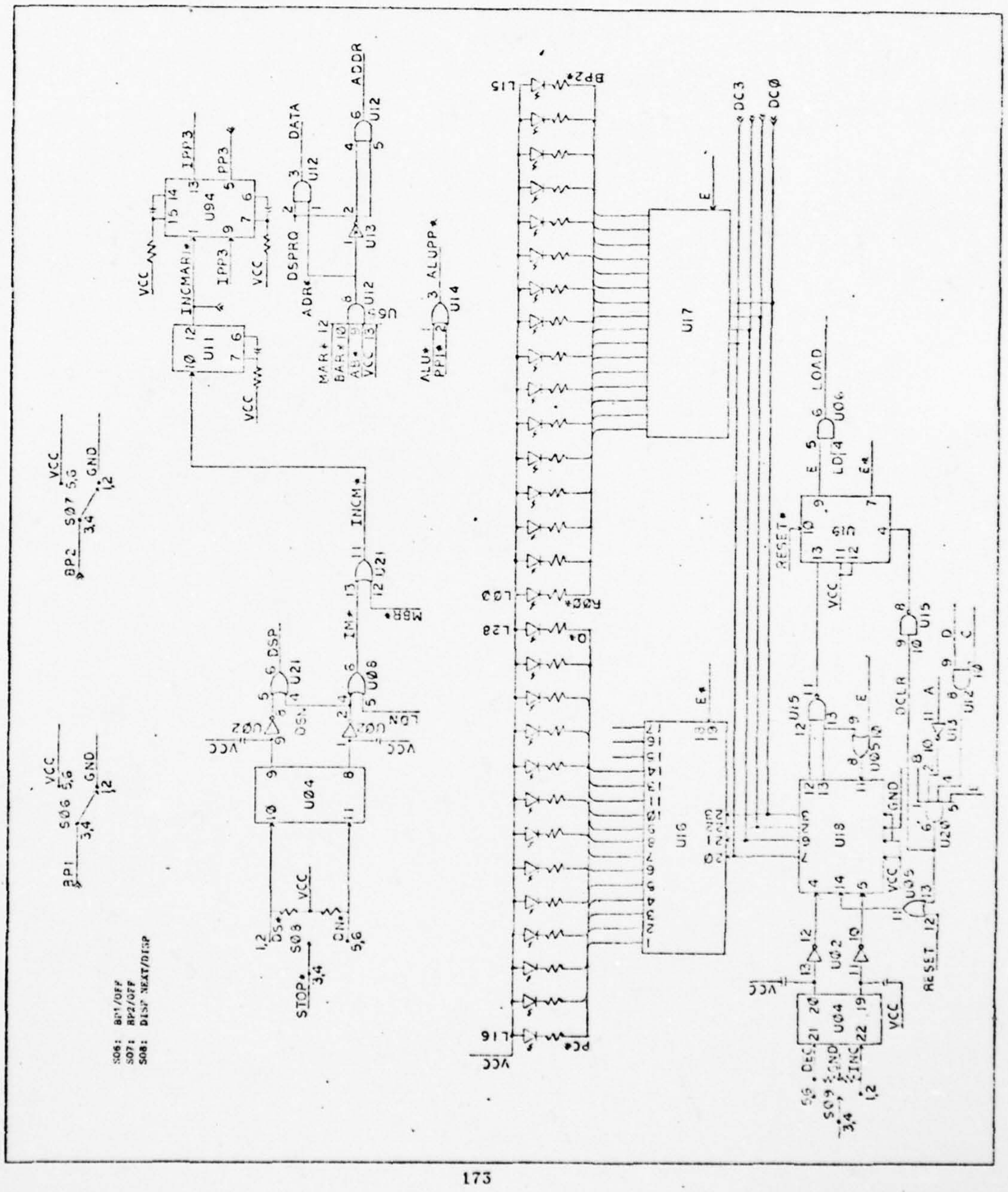


Figure A34. (Continued)

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

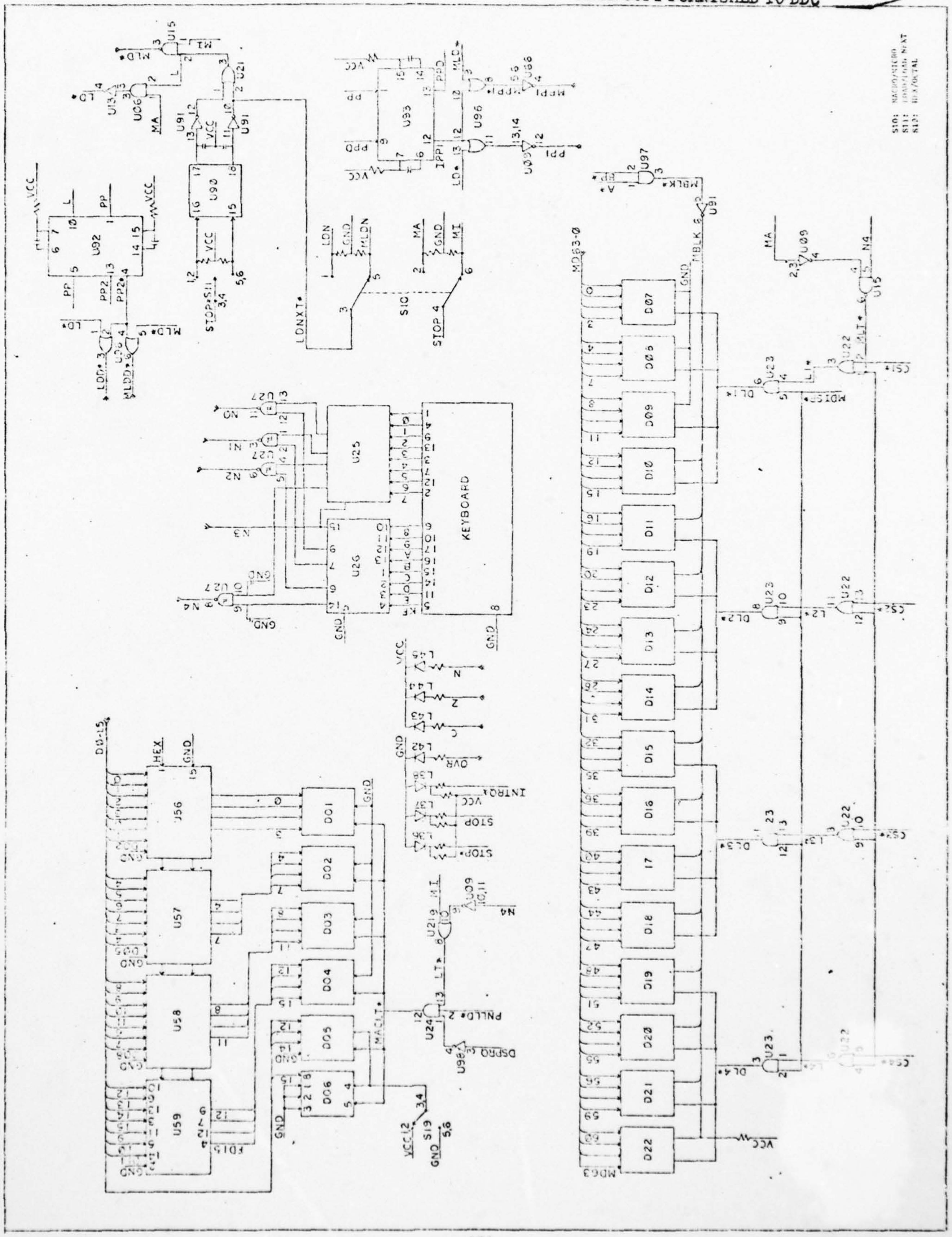


Figure A34. (Continued)

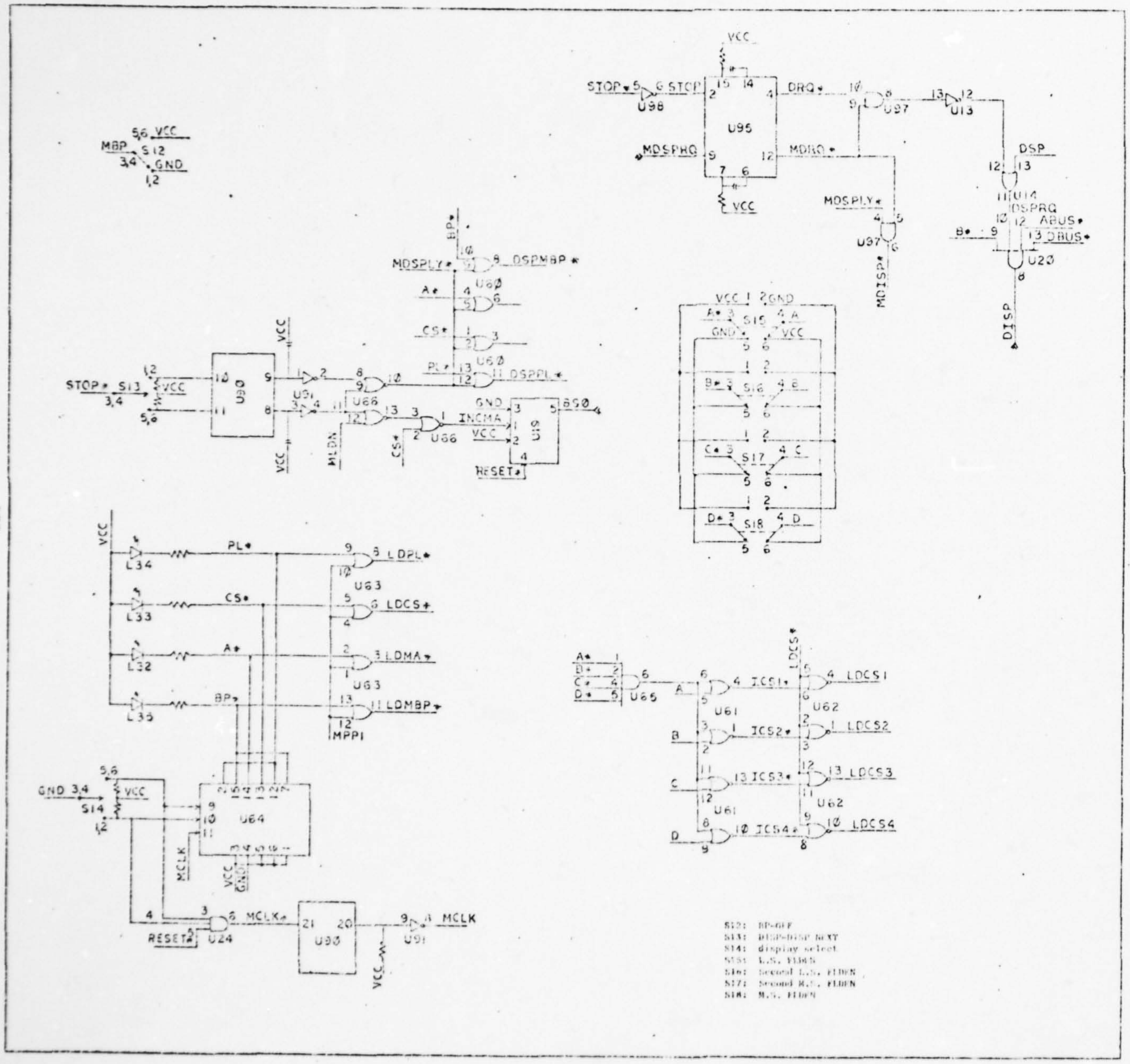
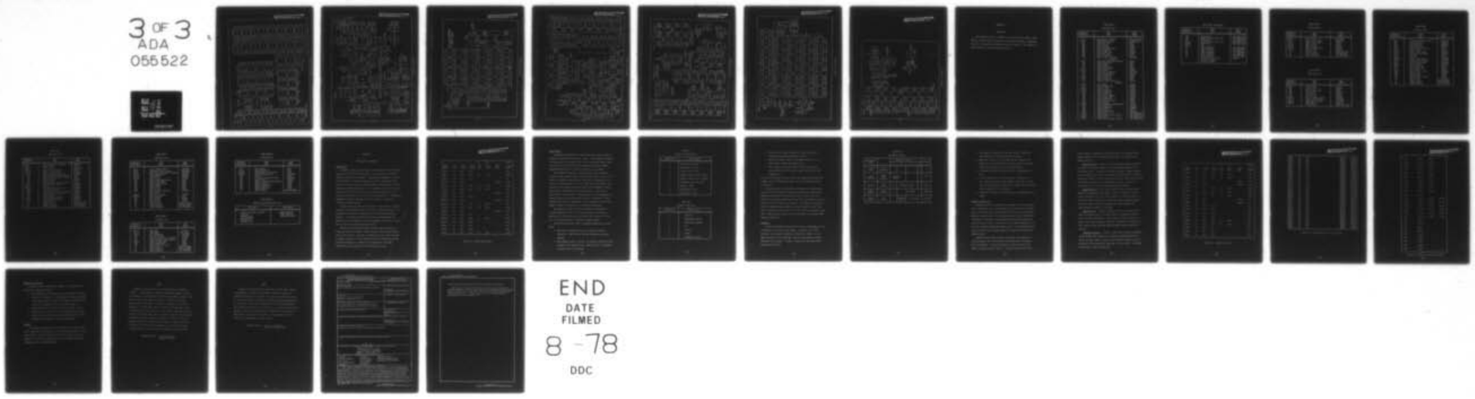


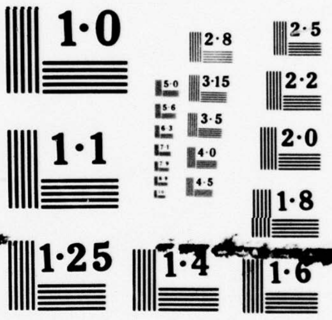
Figure A34. (Continued)

AD-A055 522 · AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 9/2
MIME: MICROPROGRAMMABLE MINICOMPUTER EMULATOR.(U)
MAR 78 R E PURVIS, R D YOHO
UNCLASSIFIED AFIT/GE/EE/78-6 NL

3 OF 3
ADA
055522



END
DATE
FILMED
8 -78
DDC


1·0
2·8
2·5
5·0
3·15
2·2
5·6
3·5
2·0
1·1
4·0
4·5
1·8
1·25
1·4
1·6

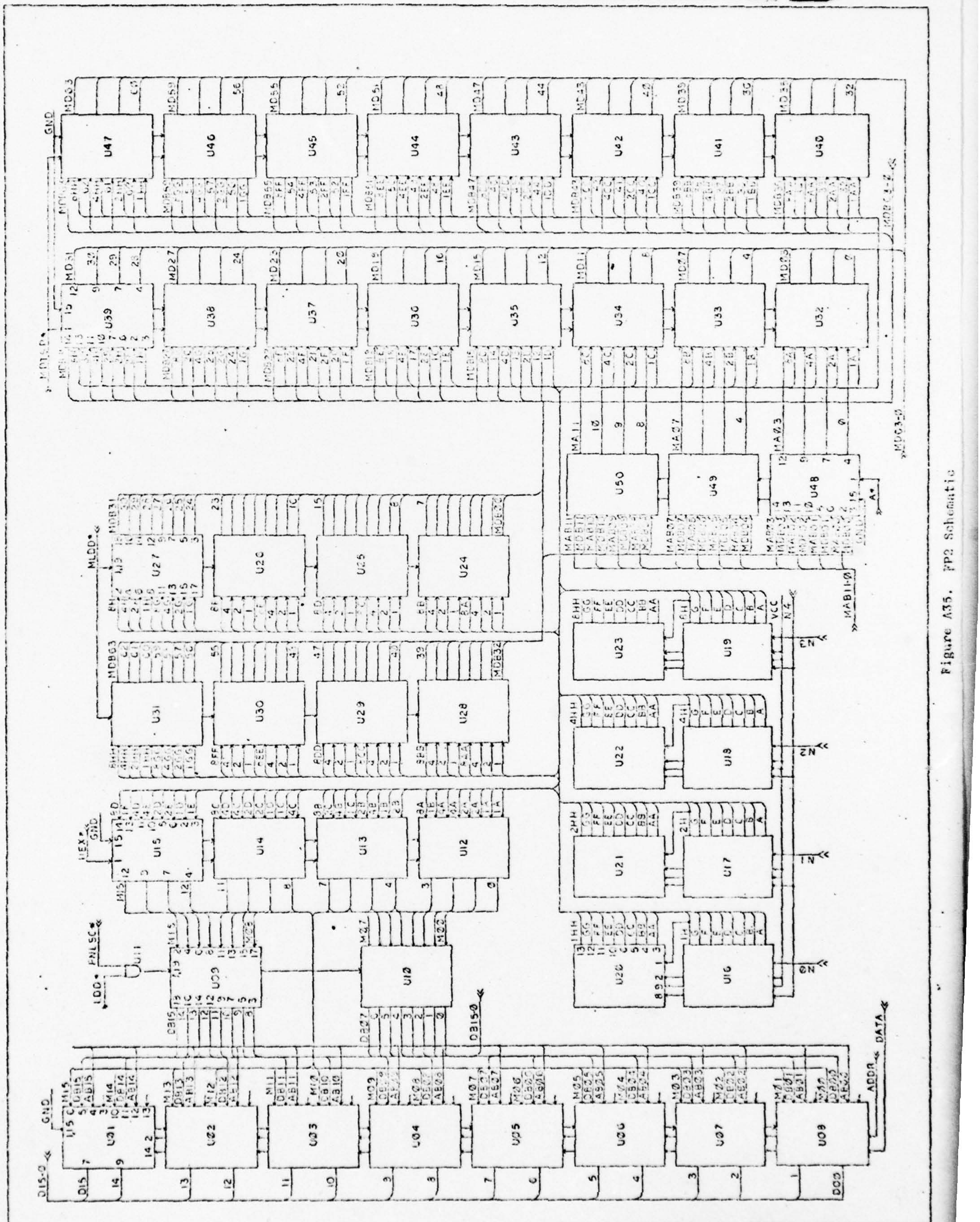


Figure A35. FP2 Schematic

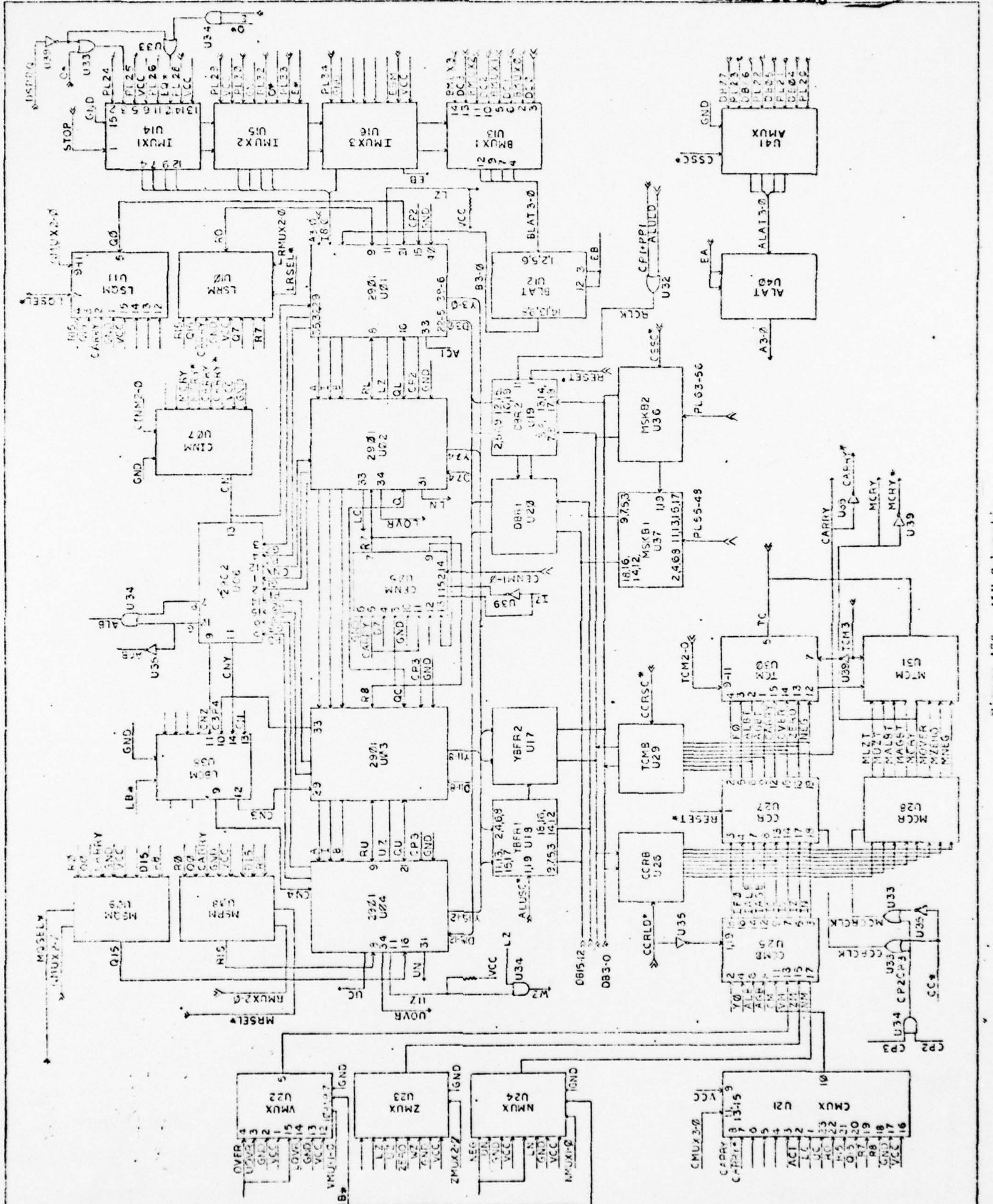


Figure A36. ALU Schematic

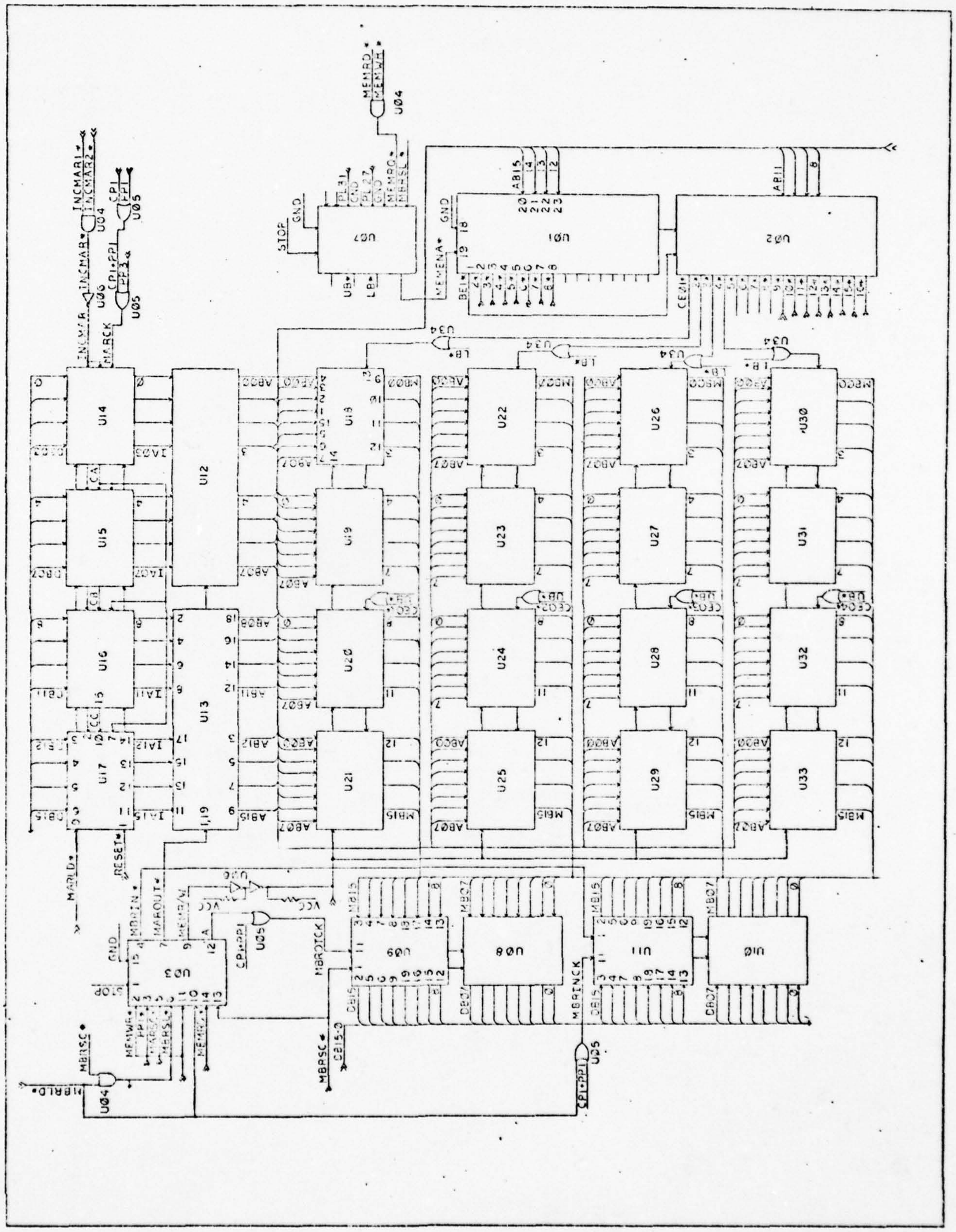


Figure A37. MEM1 Schematic

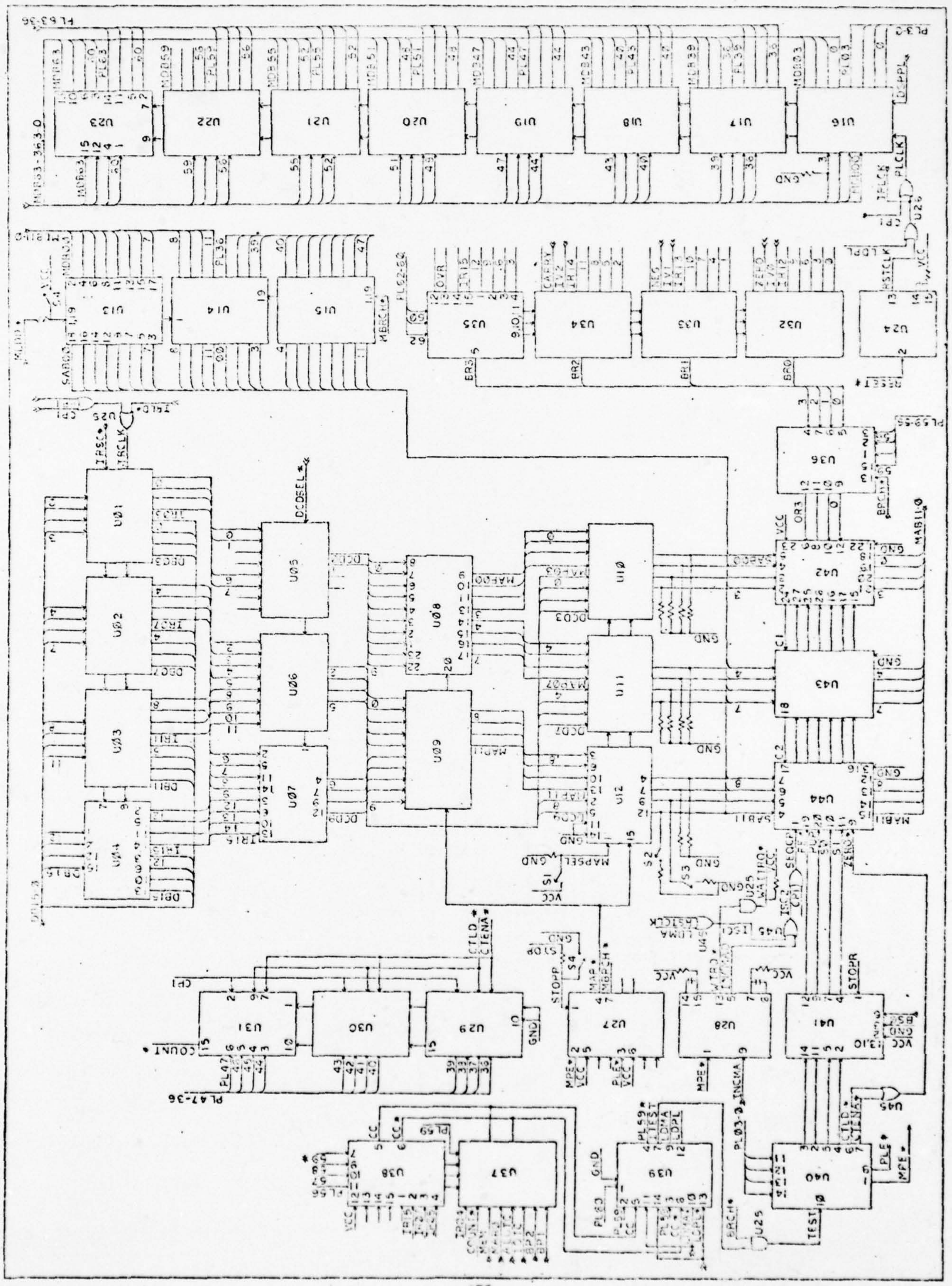


Figure A38. CCU1 Schematic

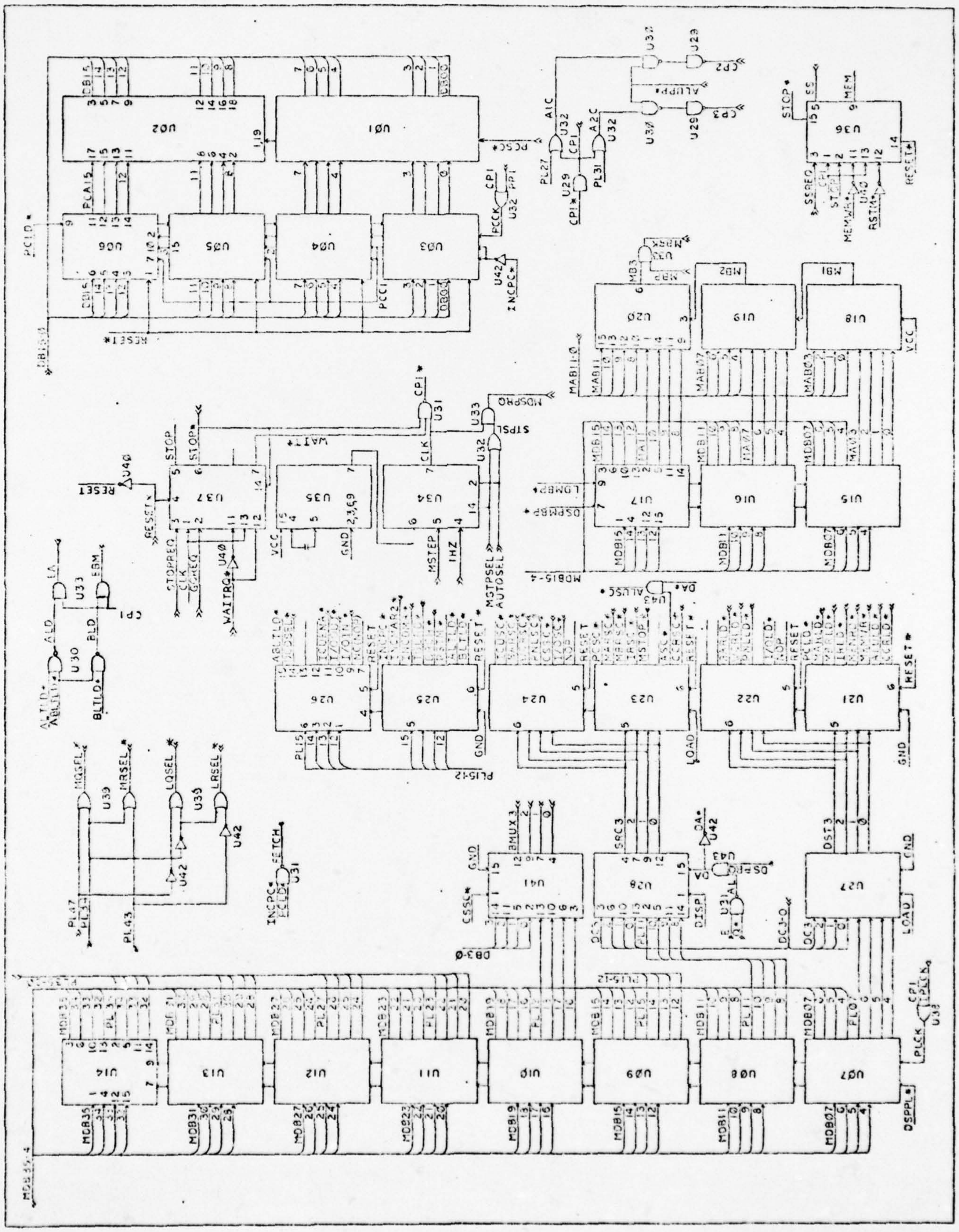


Figure A39. CCU2 Schematic

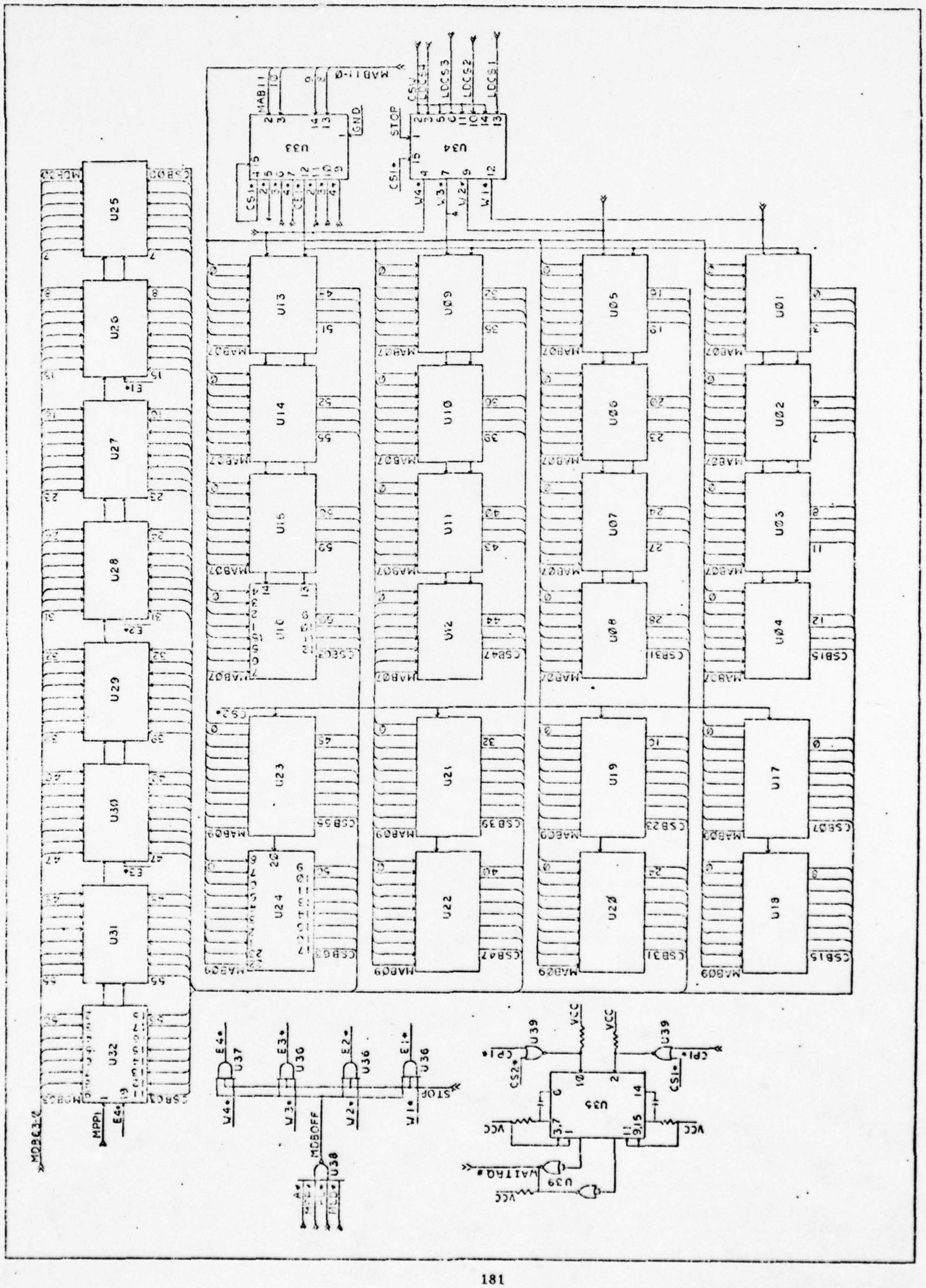


Figure A40. CS1 Schematic

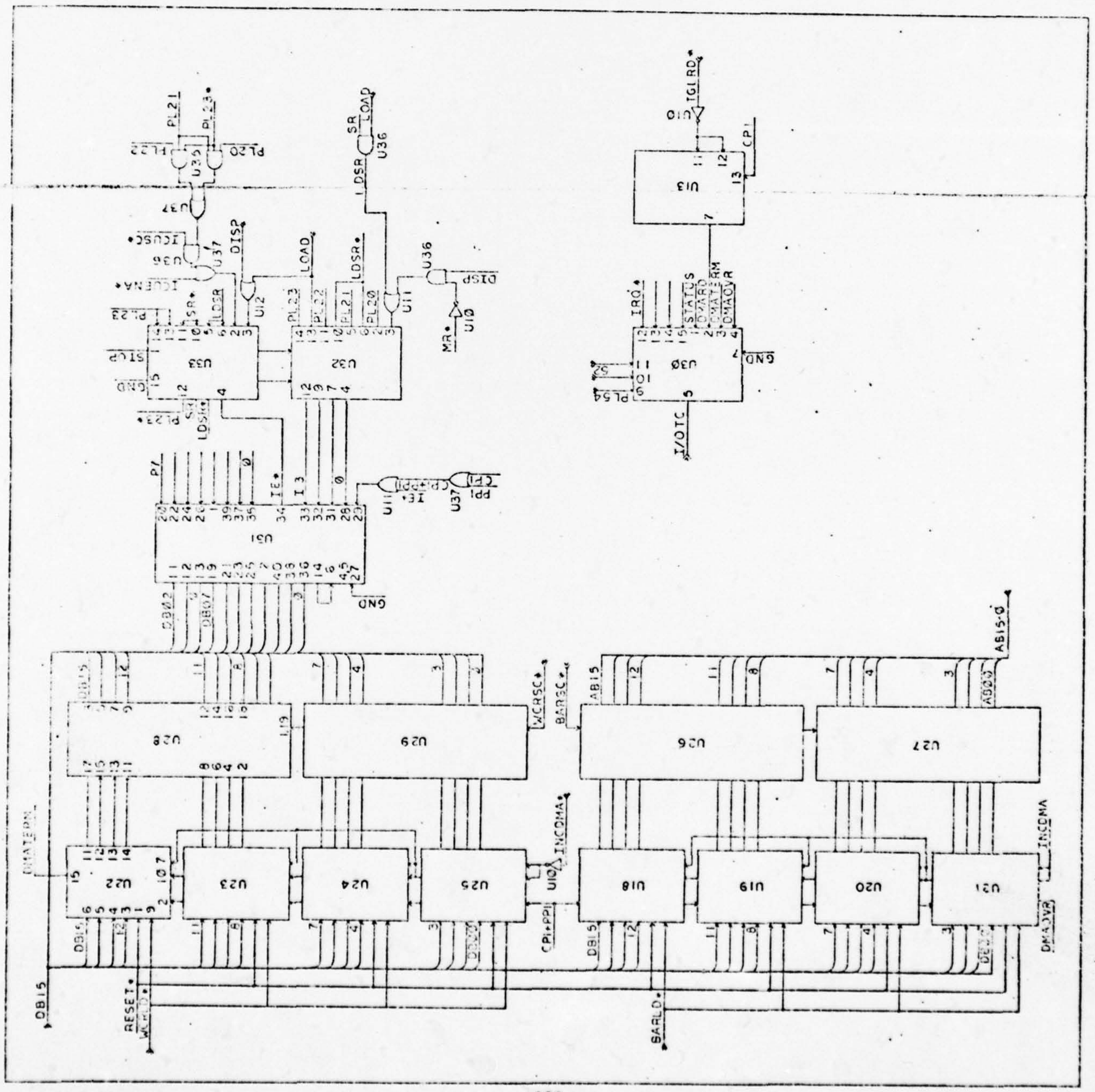


Figure A41. I/O Schematic

Appendix 2

Parts List

This appendix provides a complete list of parts used in MIME. Table AXXXIII, I/O Parts List, contains only those parts necessary to implement the basic I/O capability discussed in the User's Manual. The complete I/O design and parts list are contained in Ref. 6.

Table AXXVI

FP1 Parts List

Component Designation	Part Name	Part Number
K01	hexadecimal keyboard	Digitran KL036
D01-22	hexadcmical display	HP 5082-7340
U01	precision timer	72555
U02	hex inverter	7414N
U03	quad S-R latch	74179N
U04	8-bit bistable latch	74100N
U05	quad 2-input OR	7432N
U06	quad 2-input AND	74S108N
U07	precision timer	72555
U08	dual 4-input NOR	7425N
U09	quad 2-1 mux (inverted output)	74LS158N
U10-11	dual monostable multivibrator	54123
U12	quad 2-input AND	7408
U13	hex inverter	7414N
U14	quad 2-input OR	7432
U15	quad 2-input NAND	7400N
U16-17	4-16 decoder	74154N
U18	4-bit shift register	74LS193N
U19	dual J-K flip-flop	74S112N
U20	dual 4-input AND	74H21
U21-22	quad 2-input OR	7432N
U23	quad 2-input AND	7408
U24	triple 3-input AND (O.C. output)	54S15J
U25-26	8-3 encoder	74148N
U27	quad 2-input NAND	74132N
U56-59	quad 2-1 mux	74157N
U60	quad 2-input OR	74LS32N
U61-62	quad 2-input NOR	7402N
U63	quad 2-input OR	74LS32N
U64	4-bit shift register	74194N
U65	dual 4-input NAND	74H21N
U66	quad 2-input NOR	5402A
U90	8-bit bistable latch	74100N
U91	hex inverter	7414N
U92-95	dual monostable multivibrator	74123J
U96	quad 2-input OR	74LS32N
U97	quad 2-input AND	74LS08N
U98	hex inverter	5414N
U99	dual J-K flip-flop	74S112N
S01-2	DPST switch	C&K 7201J2-V4-1
S03-5	DPDT center off mom. switch	C&K 7205J2-V4-1
S06-7	DPST switch	C&K 7201J2-V4-1
S08-9	DPDT center off mom. switch	C&K 7201J2-V4-1

Table AXXVI (Continued)

Component Designation	Part Name	Part Number
S10	DPST switch	C&K 7201J2-V4-1
S11	momentary DPDT center off	C&K 7205J2-V4-1
S12	DPST switch	C&K 7201J2-V4-1
S13-14	momentary DPDT center off	C&K 7205J2-V4-1
S15-18	DPST switch	C&K 7201J2-V4-1
L01-43	LED	
C1	cable connector	3-M 3492-3005
C2	cable connector	3-M 3496-3005
C3	cable connector	3-M 3492-3005
C4	cable connector	3-M 3496-3005
C5	cable connector	3-M 3493-3005
C6	cable connector	3-M 3492-3005
	printed circuit board	Vector
	24 pin IC socket	
	16 pin IC socket	

Table AXXVII

FP2 Parts List

Component Designation	Part Name	Part Number
U01-8	dual 4-1 mux	54LS253DM
U09-10	octal 3-state buffer	74LS244N
U11	quad 2-input NAND	74S04N
U12-15	quad 2-1 mux	74157N
U16-23	shift register	54164J
U24-31	octal 3-state buffer	74LS244N
U32-50	quad 2-1 mux	74157N
B	circuit card	Augat 8136 WR61
C2	cable connector	3-M 3496-3005
C4	cable connector	3-M 3496-3005

Table AXXVIII

MEM1 Parts List

Component Designation	Part Name	Part Number
U01-2	4-16 decoder	74154N
U03	quad 2-1 mux	74S257N
U04	quad 2-input NAND	74S08
U05	2-input OR	74LS32N
U06	hex inverter	74S04N
U07	quad 2-1 mux	74S257N
U08-11	octal 3-state register	74LS374N
U12-13	octal 3-state buffer	74LS244N
U14-17	quad binary counter	74LS161AN
U18-33	256x4 RAM	2112A-2
G	circuit card	Augat 8136 WR61

Table AXXIX

ALU Parts List

Component Designation	Part Name	Part Number
U01-4	4-bit microprocessor slice	Am2901DC
U05	8-1 3-state mux	74LS251N
U06	lookahead carry	Am2902PC
U07-11	8-1 3-state mux	74LS251N
U12	4-bit bistable latches	7475N
U13-16	quad 2-1 mux	74157N
U17-18	octal 3-state buffer	74LS244N
U19-20	octal 3-state register	74LS273N
U21	1-16 mux	74150N
U22-24	8-1 mux	74151AN
U25-26	octal 3-state buffer	74LS244N
U27-28	octal 3-state register	74LS273N
U29	octal 3-state buffer	74LS244N
U30-31	8-1 3-state mux	74LS251N
U32-33	quad 2-input OR	7432N
U34	quad 2-input NAND	74LS08N
U35	hex inverter	7414N
U36-37	octal 3-state buffer	74LS244N
U38	quad 2-1 mux	74157N
U39	hex inverter	7414N
U40	4 bit bistable latches	74175N
U41	quad 2-1 mux	74157N
F	circuit card	Augat 8136WR61
C6	cable connector	3-M 3492-3005
C8	cable connector	3-M 3493-3005

Table AXXX.

CCU1 Parts List

Component Designation	Part Name	Part Number
U01-4	quad D register (3-state)	Am2918PC
U05-7	quad 2-1 mux	74157N
U08-9	1K x 8 EPROM	TMS 2708JL
U10-12	quad 2-1 3-state mux	74S257N
U13-15	octal 3-state buffer	74LS244N
U16-23	quad D register (3-state)	Am2918PC
U24	dual monostable multivibrator	74123N
U25	quad 2-input AND (O.C. output)	7409N
U26	quad 2-input OR	74LS32N
U27	quad 2-1 mux	74157N
U28	dual monostable multivibrator	54123N
U29-31	up/down binary counter	74LS169N
U32-35	8-1 mux	74151N
U36	16-way Branch Control Unit	Am29803DC
U37-38	8-1 mux (3-state)	74251N
U39	quad 2-1 mux (inverted output)	74LS158N
U40	Next Address Control Unit	Am29811
U41	quad 2-1 mux	74157N
U42	Microprogram Sequencer	Am2909DC
U43-44	Microprogram Sequencer	Am2911PC
U45	quad 2-input OR	74LS32N
D	circuit card	Augat 8136 WR61
C3	cable connector	3-M 3492-3005
C7	cable connector	3-M 3493-3005

Table AXXXI

CCU2 Parts List

Component Designation	Part Name	Part Number
U01-2	octal 3-state buffer	74LS244N
U03-6	synchronous 4-bit counter	74LS161AN
U07-17	quad D(3-state output) register	Am2918PC
U18-20	4-bit comparator	74LS85N
U21-26	3-8 decoder	74S138N
U27-28	quad 2-1 mux	74S257N
U29	quad 2-input NAND buffer	7437N
U30	quad 2-input NAND	7400N
U31	triple 3-input NAND	7410
U32	quad 2-input	7432N
U33	quad 2-input	7408
U34	dual 4-1 mux	74153J
U35	dual voltage controlled oscillator	74LS124N
U36-37	dual J-K flip-flop	74S112N
U38-39	quad 2-input OR	7432
U40	hex inverter	74S04N
U41	quad 2-1 mux	74S257N
U42	hex inverter	7404
E	circuit card	Augat 8136WR61
C5	cable connector	3-M 3493-3005
C8	cable connector	3-M 3493-3005

Table AXXXII

CS1 Parts List

Component Designation	Part Name	Part Number
U01-16	256X4 RAM	2112A-2
U17-24	1KX8 EPROM	TMS 2708JL
U25-32	bus transceiver	74LS245N
U33	2-4 line decoder	74S139N
U34	quad 2-1 mux (inverted output)	74158N
U35	dual monostable multivibrator	74123J
U36-37	triple 3-input NAND	74H11
U38	dual 4-input NAND	7420N
U39	quad 2-input O.C. NOR	7433N
C	circuit card	Augat 8136WR61
C1	cable connector	3-M 3492-3005
C7	cable connector	3-M 3493-3005

Table AXXXIII

I/O Parts List

Component Designation	Part Name	Part Number
U10	hex inverter	7414
U11-12	quad 2-input OR	7432
U18-25	synchronous 4-bit counter	74161
U26-29	octal 3-state buffer	74LS244N
U30	8-1 mux	74151
U31	Priority Interrupt Encoder	Am2914PC
U32	quad 2-1 mux	74157N
U33	quad 2-1 mux (inverted output)	74158N
U34	quad 2-input OR	7414N
U35	triple 3-input AND	74H11
U36	quad 2-input AND	7408
U37	quad 2-input OR	7432N
H	circuit card	Augat 8136WR61

Table AXXXIV

Chassis and Miscellaneous Parts

Part Name	Part Number
circuit card edge connector	Augat 14005-1P1
card rack	Augat 8170-R63
extender card	Augat 8136-13861
BNC connectors	
binding posts	
barrier strip	

Appendix B

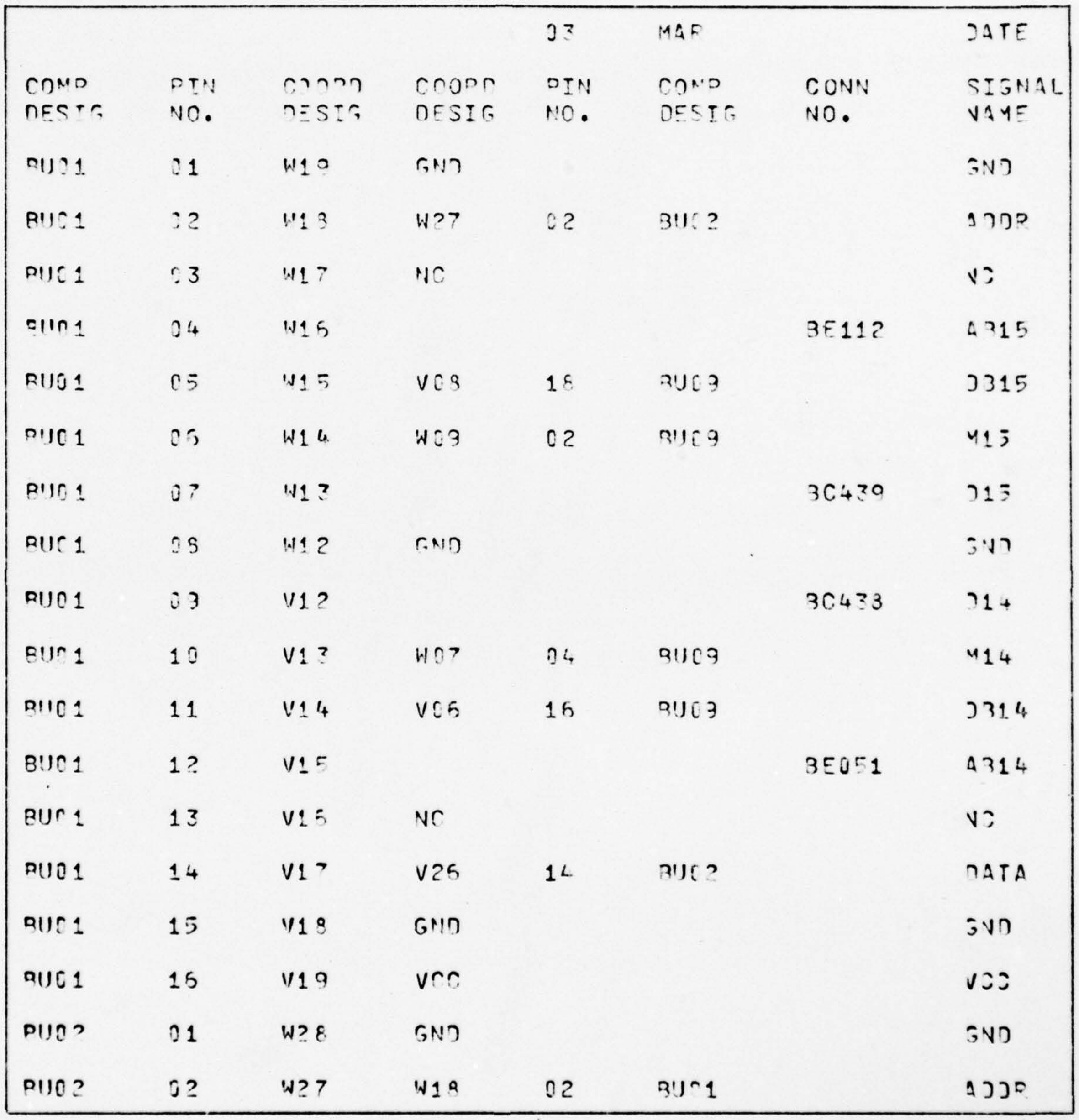
Configuration Management

Introduction

In order to manage the development of MIME and provide a definitive description at the end of the project, it was necessary to establish specific configuration procedures. Configuration management is defined by MIL STD 483 as a discipline applying technical and administrative direction and surveillance to document product characteristics, control changes, and update documentation. This appendix specifies the various procedures used during this project and provides explanation of the notation and conventions used in the system documentation. Configuration management procedures were applied to the project in two phases: informal and formal control.

During the requirements definition phase of the project, informal procedures and documentation were sufficient to maintain control. This documentation consisted of requirements lists and functional block diagrams. The informal procedures consisted of merely dating each document as it was completed so that the documents with the most recent date reflected the current system definition.

Because the configuration became much more detailed during the design and fabrication phases, formal procedures and detailed documentation were required. Two types of documentation were used to provide a complete description of the system electrical configuration: wiring tables and schematics. Because of their importance, the formal documentation and procedures will be discussed in detail.



				03	MAR		DATE
COMP DESIG	PIN NO.	COORD DESIG	COORD DESIG	PIN NO.	COMP DESIG	CONN NO.	SIGNAL NAME
BU01	01	W19	GND				GND
BU01	02	W18	W27	02	BU02		ADDR
BU01	03	W17	NC				NC
BU01	04	W16				BE112	AB15
BU01	05	W15	V08	18	BU09		DB15
BU01	06	W14	W09	02	BU09		M15
BU01	07	W13				BC439	D15
BU01	08	W12	GND				GND
BU01	09	V12				BC438	D14
BU01	10	V13	W07	04	BU09		M14
BU01	11	V14	V06	16	BU09		DB14
BU01	12	V15				BE051	AB14
BU01	13	V16	NC				NC
BU01	14	V17	V26	14	BU02		DATA
BU01	15	V18	GND				GND
BU01	16	V19	VCC				VCC
BU02	01	W28	GND				GND
BU02	02	W27	W18	02	BU01		ADDR

Figure B1. Wiring Table Sample

Wiring Tables

The physical configuration of MIME consisted of various types of components mounted on circuit cards. Thus, it was required to document the connections on each card as well as the connections between cards. A portion of the ALU wiring which was generated using the computer programs discussed later is shown in Figure B1 as an example.

Each component was given a unique identifier that designated component type, component number, and the circuit card on which it was located. In addition, each possible location of a component pin on a circuit card corresponded to a card column and row coordinate. The location of the component pins thus specifies the component location and provides a description of the physical layout. Each circuit card connector was also given a unique identifier that designated the connector type and the circuit card on which it was located. These component and connector designations used in conjunction with the pin numbers allowed all connections to be completely described.

Associated with each pin of both connectors and components is a unique alphanumeric name. In so far as possible, this name is descriptive of the signal type and function that is present on that pin. As asterisk was used to indicate a negated signal.

The specific guidelines used in assigning designators are given below:

1. Each circuit card shall be given a unique on character identifier. Table BI lists the card designators and their meaning.
2. Each component shall be given a one character identifier which designates the component type. Table BII lists the component designators and their meaning.

Table BI

Circuit Card Identifiers

Identifiers	Circuit Board
A	Control Store 2 (CS2)
B	Front Panel 2 (FP2)
C	Control Store 1 (CS1)
D	Computer Control Unit 1 (CCU1)
E	Computer Control Unit 2 (CCU2)
F	Arithmetic Logic Unit (ALU)
G	Memory 1 (MEM1)
H	Input/Output (I/O)
P	Front Panel 1 (FP1)

Table BII

Component Identifiers

Identifiers	Component Type
C	capacitor
D	hexadecimal display
K	hexadecimal keyboard
L	LED
R	resistor
S	Switch
U	integrated circuit

3. Within a circuit card, components of each type shall be consecutively numbered beginning with one.

4. Each card connector shall be designated either E or C to denote edge or cable type respectively.

5. The coordinate system provided on the wire wrap circuit cards shall be used to specify a unique location for each component pin.

The various designators and their format are summarized in Table BIII. A sample of how the designators appear in a wiring table was shown in Figure B1.

The primary type of wiring table used during the project was an intra circuit card table that completely described the component-to-component and component-to-connector wiring on a given card. One table of this type was generated for each card, and Figure B1 contains a portion of the ALU table as an example. This table is presented in order of component and pin numbers; each entry identifies the items that are connected, the location of the wire, and the associated signal name. Special purpose formats of the wiring table are discussed under Computer Program Aids.

Schematics

Circuit card schematics were used to provide a diagrammatical representation of the circuit card wiring. A complete set of MIME schematics are included in Appendix A, MIME User's Manual. Due to space limitations on the drawings, only the signal names of primary importance are included. In order to simplify the drawings several conventions were used:

Table BIII

Designator Formats

Designator Type		Character Position				Example
	1	2	3	4	5	
circuit card	card designator					B
card coordinate	card column (letter)	card row (number)				W43
component	card location	type	component number			BU01
edge connector	card location	E	pin number			BE106
cable connector	card location	C	connector number	pin number		BC207

1. Even though the logic gates used were on IC's, individual logic symbols were used for functional clarity.
2. Where two or more IC's of the same type are adjacent, pin numbers for the connections are given for only one, and the others may be assumed to follow the same pattern.
3. A connection between intersecting lines is designated by a dot.
4. A fillet connecting two lines indicates a connection to a bus. Where an IC has several adjacent connections to a bus, only the first and last connections are named, and intervening ones are assumed to be consecutive.
5. Connections to edge or cable connectors are designated by the symbol.

Computer Program Aids

Automated aids for the generation and update of the wiring tables were a significant factor in providing accurate and versatile documentation. The first step of the documentation process was to manually generate a description of the wiring on each card in the format shown in Figure B1. This data was then keypunched and stored on permanent disk files (Ref 23:6-1). From this point, all further manipulation of the files could be accomplished interactively. This allowed convenient access to the files for making changes and automatically generating updated tables.

Because of the volume of each wiring table, the likelihood of error was great, and it was desired to have some means of assuring their consistency. It was also desired to be able to present the tables in special purpose formats. In order to achieve these goals several computer programs were written and these are discussed below. Complete details of these programs may be found in a report by the authors (Ref 24).

Consistency Check. Because the wiring connections were described for each pin of every component, each wire actually appeared in the wiring table twice. Therefore, the authors devised a computer program to verify that the two entries describing a single wire were consistent. Inconsistent entries or those for which no mate was found were identified on a computer printout. This information was used to correct the wiring table until internal consistency had been achieved.

Alphabetized Sort. The initial format of the wiring tables was indexed by component and pin numbers. However, frequently the need arose to determine the circuit card location of a specific signal. To facilitate signal tracing during troubleshooting, a computer program to sort the table in order to alphabetized signal name was used. Figure B2 shows an example of this data.

Connector Sort. A computer program was devised to extract and order all connector entries from each table. The separate connector lists were then concatenated to form a master connector pin list. A sample of the data is shown in Figure B3. Presentation of the data in this format was used when only signals between circuit cards were of interest.

Backplane Generator. In order to facilitate wiring the backplane (wiring between edge connectors), a computer program was devised to extract the edge connector entries from the master connector pin list and match all those having the same name to form an entry. An example of this data is shown in Figure B4.

				C3	MAP		DATE
COMP DESIG	PIN NO.	COORD DESIG	COORD DESIG	PIN NO.	COMP DESIG	CONN NO.	SIGNAL NAME
BU02	02	W27	W18	02	BU01		ADDR
BU01	02	W18	W27	02	BU02		ADDR
BU01	12	V15				BE051	AB14
BU01	04	W16				BE112	AB15
BU01	14	V17	V26	14	BU02		DATA
BU01	11	V14	V06	16	BU09		DB14
BU01	05	W13	V08	18	BU09		DB15
BU01	09	V12				BC436	D14
BU01	07	W13				BC439	D15
BU01	15	V18	GND				GND
BU02	01	W28	GND				GND
BU01	01	W19	GND				GND
BU01	08	W12	GND				GND
BU01	10	V13	W07	04	BU09		M14
BU01	06	W14	W09	02	BU09		M15
BU01	13	V16	NC				NC
BU01	03	W17	NC				NC
BU01	16	V19	VCC				VCC

Figure B2. Alphabetized Data

BU24	03	B08	BE002	MDB00
BU24	07	B04	BE003	MDB02
BU24	12	A02	BE004	MDB04
BU24	16	A06	BE005	MDB06
BU25	03	D08	BE006	MDB08
BU25	07	D04	BE007	MDB10
BU25	12	C02	BE008	MDB12
BU25	16	C06	BE009	MDB14
BU26	03	F08	BE010	MDB16
BU26	07	F04	BE011	MDB18
BU26	12	E02	BE012	MDB20
BU26	16	E06	BE013	MDB22
BU27	03	H08	BE014	MDB24
BU27	07	H04	BE015	MDB26
BU27	12	G02	BE016	MDB28
BU27	16	G06	BE017	MDB30
BU28	03	K08	BE018	MDB32
BU28	07	K04	BE019	MDB34
BU28	12	J02	BE020	MDB36
BU28	16	J06	BE021	MDB38
BU29	03	M08	BE022	MDB40
BU29	07	M04	BE023	MDB42
BU29	12	L02	BE024	MDB44
BU29	16	L06	BE025	MDB46
BU30	03	P08	BE026	MDB48
BU30	07	P04	BE027	MDB50
BU30	12	N02	BE028	MDB52
BU30	16	N06	BE029	MDB54
BU31	03	S08	BE030	MDB56
BU31	07	S04	BE031	MDB58
BU31	12	R02	BE032	MDB60
BU31	16	R06	BE033	MDB62
BU13	01	M27	BE035	HEX*
BU48	02	B26	BE036	MAB00
BU48	11	A22	BE037	MAB02
BU49	02	D26	BE038	MAB04
BU49	11	C22	BE039	MAB06
BU50	02	F26	BE040	MAB08
BU50	11	E22	BE041	MAB10
BU11	05	H22	BE042	PNLSC*

Figure B3. Master Connector Pin List Excerpt

AB00	BE044	GE044	
AB01	BE105	GE105	
AB02	BE045	GE045	
AB03	BE106	GE106	
AB04	BE046	GE046	
AB05	BE107	GE107	
AB06	BE047	GE047	
AB07	BE108	GE108	
AB08	BE048	GE048	
AB09	BE109	GE109	
AB0	HE044		
AB10	BE049	GE049	HE049
AB11	BE110	GE110	HE110
AB12	BE050	GE050	HE050
AB13	BE111	GE111	HE111
AB14	BE051	GE051	HE051
AB15	BE112	GE112	HE112
AB1	HE105		
AB2	HE045		
AB3	HE106		
AB4	HE045		
AB5	HE107		
AB6	HE047		
AB7	HE108		
AB8	HE048		
AB9	HE109		

Figure B4. · Backplane Wiring Table Excerpt

Management Procedures

In order to control documentation changes, it was necessary to specify the following procedures:

1. Upon completion, each document was given an effectivity date.
2. Changes to a document were accomplished by noting the changes in red and entering the date of the change. The effectivity date was also changed to that of the modification.
3. Upon accumulation of sufficient red-line changes to cause confusion, a revised version of the document was generated.
4. Each circuit card was physically dated with the date of the document(s) describing its current configuration.

Summary

This appendix has described the procedures used by the authors to achieve and maintain configuration control during the project. While the procedures could have been more formal and rigorous, they were adequate to describe the configuration, control changes, and update documentation in a systematic manner.

Vita

Richard E. Purvis was born on 13 October 1946 in Portsmouth, Virginia. A 1965 graduate of Hampton High School, Hampton, Virginia, he attended Virginia Polytechnic Institute, Blacksburg, Virginia, from which he received the degree of Bachelor of Science (Electrical Engineering) in June, 1970. He received a commission in the USAF through OTS in March 1971, and was assigned to Williams AFB, Arizona for Undergraduate Pilot Training. He received his aeronautical rating of Pilot in February 1972, and was assigned to Shaw AFB, South Carolina for upgrade training in the RF-4C Phantom II aircraft. He was subsequently assigned to Zweibrucken AB, Germany in October 1972, where he served as an RF-4C aircraft commander until entering the School of Engineering, Air Force Institute of Technology, in August 1976.

Permanent Address: 117 Lynnhaven Drive
Hampton, VA 23666

Vita

Ronald D. Yoho was born on 30 March 1951 in Evansville, Indiana. A 1969 graduate of Cannelton High School, Cannelton, Indiana, he attended the University of Evansville, Evansville, Indiana from which he received the degree of Bachelor of Science (Electrical Engineering) in June, 1973. He received a commission in the USAF through AFROTC and reported to active duty on 17 June 1973. He served as an Electrical Engineer in the Tactical LORAN Systems Program Office at Hanscom AFB, Massachusetts until entering the School of Engineering, Air Force Institute of Technology, in August of 1976.

Permanent Address: 714 St. Louis Avenue
Cannelton, Indiana 47520

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GE/EE/78-6 / 2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) MIME: MICROPROGRAMMABLE MINICOMPUTER EMULATOR	5. TYPE OF REPORT & PERIOD COVERED MS Thesis
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) RICHARD E. PURVIS, Capt, USAF; RONALD D. YOHO, Capt, USAF	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT/EN) Wright-Patterson Air Force Base, Ohio 45433	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS	12. REPORT DATE March 1978
	13. NUMBER OF PAGES 207
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report) UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES Approved for public release; IAW AFR 190-17

[illegible]

JERRAL F. GUESS, Capt, USAF
Director of Information

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Am2900	emulation	computer control
microprogramming	microprocessor	microprogrammed control
bit slice	microcomputer	generalized architecture
microprogrammable	minicomputer	
emulator	microword	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report addresses the design and fabrication of a microprogrammable, general purpose minicomputer specifically configured for use in an educational environment. In order to have utility as an instructional aid in the areas of computer control and microprogramming, it was determined that such a machine must have the following attributes: educationally oriented human interface, educationally oriented design, and user-microprogrammability. These high level attributes were then used as a basis for deriving functional and detail requirements. The Am2900 Bipolar Microprocessor Family was used as the basis for

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

realizing the design which satisfied the defined requirements.

The project resulted in hardware which was used by students in digital engineering classes and laboratory to investigate microprogramming and its application to emulation. This report includes representative microcode necessary to emulate sample PDP-11/03 instructions as a demonstration of MIME capabilities.